ALGORITMOS DISTRIBUÍDOS PARA ESCALONAMENTO LIVRE DE COLISÕES E SINCRONIZAÇÃO DE RELÓGIOS EM REDES DE SENSORES SEM FIO

André da Costa Pinho

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Felipe Maia Galvão França
Daniel Ratton Figueiredo

Rio de Janeiro
Dezembro de 2011

ALGORITMOS DISTRIBUÍDOS PARA ESCALONAMENTO LIVRE DE COLISÕES E SINCRONIZAÇÃO DE RELÓGIOS EM REDES DE SENSORES SEM FIO

André da Costa Pinho

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Felipe Maia Galvão França, Ph.D.

Prof. Daniel Ratton Figueiredo, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Luís Felipe Magalhães de Moraes, Ph.D.

Prof. Célio Vinícius Neves de Albuquerque, Ph.D.

Prof. Katia Obraczka, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2011

Pinho, André da Costa

Algoritmos Distribuídos para Escalonamento Livre de Colisões e Sincronização de Relógios em Redes de Sensores sem Fio/André da Costa Pinho. – Rio de Janeiro: UFRJ/COPPE, 2011.

XV, 107 p.: il.; 29, 7cm.

Orientadores: Felipe Maia Galvão França
Daniel Ratton Figueiredo

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 102 – 107.

1. MAC. 2. WSN. 3. distance-2. I. França, Felipe Maia Galvão *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Este trabalho é dedicado à minha esposa Marcela e aos meus filhos Bruno, Pedro e Beatriz. À Marcela pelo apoio, amparo, consideração e amor que me deram sustentação para nunca esmorecer. Aos meus filhos, pela compreensão de minhas faltas, pelo amor, pelos sorrisos e por me ensinarem a arte de explicar problemas complexos com simplicidade. À minha família, meu tesouro, muito obrigado.

Agradecimentos

Agradeço aos meus pais, João e Isabel, pelo amor e apoio ao longo de todos estes anos.
À minha irmã Valéria pela confiança e amizade.
Aos meus orientadores, Felipe e Daniel, pela paciência, confiança e amizade.
Ao Cel Castañon, pelo apoio e amizade
Ao amigo Alexandre, pelo apoio e amizade
Aos amigos do LAM, pelo apoio e pelo papo descontraído
Aos amigos do CTEx.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ALGORITMOS DISTRIBUÍDOS PARA ESCALONAMENTO LIVRE DE COLISÕES E SINCRONIZAÇÃO DE RELÓGIOS EM REDES DE SENSORES SEM FIO

André da Costa Pinho

Dezembro/2011

Orientadores: Felipe Maia Galvão França
Daniel Ratton Figueiredo

Programa: Engenharia de Sistemas e Computação

Este trabalho propôs algoritmos distribuídos, independentes de topologia de conexão e com baixo custo, em termos de energia, para a rede. Em particular foram desenvolvidos e avaliados dois algoritmos para escalonamento de enlaces e um algoritmo para sincronização de relógios.

Os algoritmos de escalonamento foram baseados na coloração de grafos à distância-2. Particularmente, o algoritmo $Edge^3 - Sched$ realizou a coloração de arestas diretamente enquanto o $Node^2 - Sched$ realizou a coloração de arestas, por meio da coloração de nós. Estes algoritmos apresentaram uma relação de compromisso entre o número médio de cores utilizadas e o número médio de mensagens transmitidas.

No que tange a sincronização, o algoritmo proposto (RGCS), baseou-se na propriedade gradiente, na qual a sincronização de relógios se dá em função da distância entre os nós sensores, ou seja, quanto mais próximos, mais bem sincronizados. Este algoritmo mostrou ser de fácil implementação e apresentou excelentes resultados, nos diversos cenários onde foi avaliado. Mais ainda, em função do acurado ajuste de taxa, o RGCS exigiu baixíssima frequência de manutenção da sincronização, refletindo baixíssimo custo em termos de transmissão de mensagens. Estas propriedades tornaram plenamente viável a alternativa de um modelo de comunicação TDMA nas RSSF.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

DISTRIBUTED ALGORITHMS FOR COLLISION FREE SCHEDULING AND CLOCK SYNCHRONIZATION IN WIRELESS SENSOR NETWORKS

André da Costa Pinho

December/2011

Advisors: Felipe Maia Galvão França
Daniel Ratton Figueiredo

Department: Systems Engineering and Computer Science

This work has proposed distributed algorithms topology-independent and low cost in terms of energy for WSN. In particular two algorithms for link scheduling and one for clock synchronization were developed and evaluated .

The scheduling algorithms were based on graph distance-2 coloring . Particularly, $Edge^3 - Sched$ algorithm has performed edge coloring directly while $Node^2 - Sched$ algorithm did, by node coloring. These algorithms present a compromise between average number of colors used and average number of messages transmitted.

Regarding synchronization, the proposed algorithm (RGCS), was based on gradient property, in which clock synchronization is a function of the distance between the sensor nodes, ie, as closer, better synchronized. This algorithm has proved easy implementation and has shown excellent results in various scenarios where it was evaluated. Moreover, due to the accurate rate adjustment, the RGCS required very low frequency maintenance, reflecting very low cost in terms of messaging. These properties become the TDMA alternative fully feasible for WSN.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

CPU	*Central Process Unit* - Unidade de processamento central, p. 6
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance* - Múltiplo acesso ao canal com monitoração da portadora para evitar de colisão, p. 13
CSMA/CD	*Carrier Sense Multiple Access with Collision Detection* - Múltiplo acesso ao canal com monitoração da portadora para detecção de colisão, p. 9
CSMA	*Carrier Sense Multiple Access* - Múltiplo acesso ao canal com monitoração da portadora, p. 7
EML	Erro Médio Global - Erro médio entre o relógio de um nó sensor e os relógios de seus vizinhos, p. 82
FIFO	*First In First Out* - É uma estrutura de dados, tipo fila, que apresenta o seguinte critério: o elemento a ser retirado é sempre o mais antigo da estrutura, p. 90
MAC	*Medium Access Control*- Controle de acesso ao meio, p. 6
PA	Ponto de Acesso, p. 19
PID	Referência a um controlador de processos com as componentes **P**roporcional, **I**ntegradora e **D**erivativa, p. 67
RGCS	*A Robust Gradient Clock Synchronization Algorithm for Wireless Sensor Networks* - Algoritmo de sincronização de relógios proposto neste trabalho, p. 45
RSSFs	redes de sensores sem fio, p. 1
TDMA	*Time Division Multiple Access* - Múltiplo acesso ao canal por divisão do tempo, p. 3

Capítulo 1

Introdução

1.1 Redes de Sensores sem Fio

As Redes de Sensores sem Fio (RSSFs) têm chamado atenção da comunidade científica face às inúmeras possibilidades de aplicação e desafios tecnológicos, dentre as quais se destacam: a monitoração de áreas afetadas por desastres naturais ou catástrofes, a monitoração de ecossistemas, a monitoração de ambientes hostis e insalubres, a vigilância de áreas militares, o monitoramento da segurança de plantas industriais e de sistemas de infraestrutura etc. Os benefícios potenciais advindos da implantação desta tecnologia incluem: a redução da ocorrência de falhas catastróficas, o controle e conservação de recursos naturais, a aumento da produtividade industrial, a otimização de sistemas de transportes e a redução do tempo de resposta às emergências demandadas. As RSSFs são compostas por um grande número de pequenos sensores de baixo custo, baixo consumo de energia, multi-funcionais, com capacidade de sensoreamento, processamento de dados e comunicação rádio (AKYILDIZ *et al.* (2002)). Tal pesquisa é norteada pelos avanços da microeletrônica, em especial, pelos sistemas de computação miniaturizados e sistemas embarcados. KAHN *et al.* (2000) propuseram a aglutinação de um ou mais sensores, juntamente com um microcontrolador e um rádio, formando uma entidade denominada *nó sensor*. Múltiplos nós poderiam se auto-organizar em uma rede sem fio e levar a informação sobre o seu ambiente para uma determinada aplicação.

A arquitetura típica de um nó sensor é apresentada na Figura 1.1. O subsistema de energia é responsável pelo funcionamento dos outros subsistemas e é tipicamente composto por uma bateria ou um capacitor. Estas fontes de energia não renováveis têm, em grande parte, motivado inúmeras pesquisas no campo das RSSFs. Desses estudos surgiram alternativas de fontes renováveis baseadas na capacidade de coletar a energia do ambiente circundante, por meio da luz solar ou de vibrações mecânicas (SEAH *et al.* (2009)).

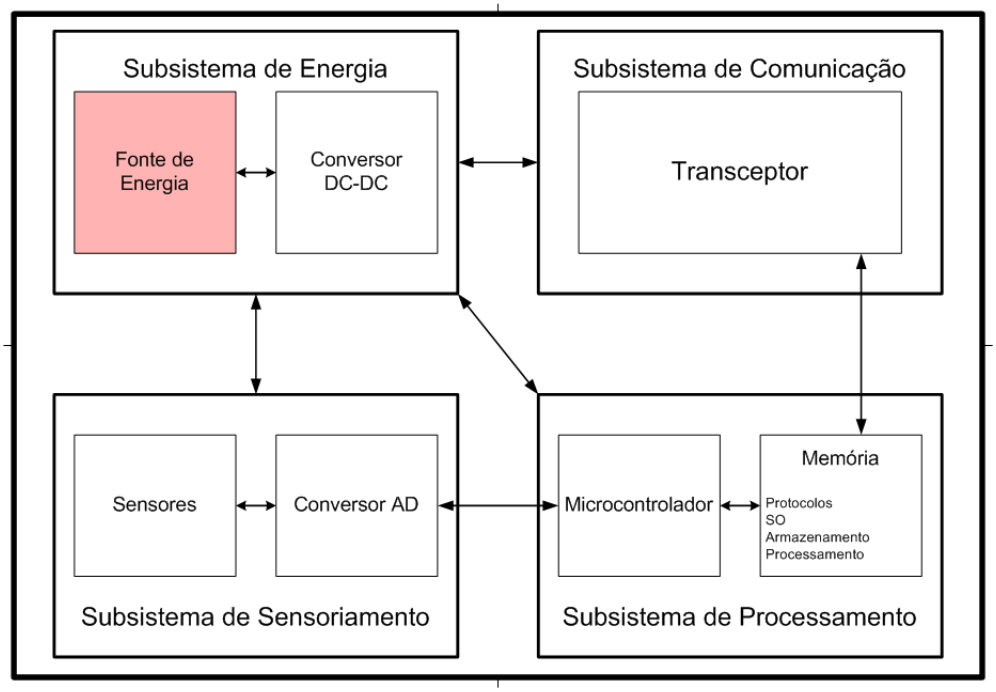

Figura 1.1: Arquitetura típica de um nó sensor

O subsistema de sensoreamento exerce o papel principal de um nó sensor, já que contém os sensores responsáveis pela coleta de informações do ambiente. O consumo de energia deste subsistema depende fortemente da natureza dos sensores, bem como da frequência de coleta das amostras. Para sensores de temperatura, a energia consumida é bastante baixa, já que o intervalo de amostragem pode ser da ordem de horas, enquanto que para o sensores multimídia, quando ativos, amostras com alto consumo de energia têm de ser tomadas, pelo menos, uma vez por segundo.

O subsistema de processamento comporta um micro-controlador, bem como a memória para suportar o sistema operacional, armazenamento e processamento de dados, e a pilha de protocolos de comunicação. Alguns estudos demonstram que a computação consome muito menos energia do que o subsistema de comunicação (POTTIE e KAISER (2000)).

O subsistema de comunicação consiste no transceptor sem fio e define a principal restrição de energia. Um dos parâmetros mais importantes a ser considerado em relação ao consumo de energia deste subsistema é seu ciclo de trabalho, ou seja, a fração de tempo em que o transceptor fica em cada um dos três estados possíveis: transmitindo, recebendo ou desligado. Mais precisamente, o subsistema de comunicação não deveria influenciar a camada superior. No entanto, a necessidade de que uma rede com recursos limitados opere com elevada eficiência energética implicou em uma violação intencional do modelo de referência de comunicação. Esta nova abordagem é chamada de *cross-layer design*. Técnicas *cross-layer* objetivam, particularmente, a redução do consumo de energia, que é claramente uma das métricas

mais importantes do desempenho de RSSFs.

1.1.1 Classificação das Redes de Sensores sem Fio

O projeto de uma RSSF é completamente dependente da área de aplicação e seus objetivos. Apesar do caráter específico de cada implementação de uma RSSF, a maior parte delas compartilha uma série de características e restrições que definem tal área de pesquisa:

1. As RSSFs precisam funcionar por longos períodos de tempo para garantir a viabilidade econômica. Tal requisito impõe severas restrições no consumo de energia (YE e HEIDEMANN (2003)), uma vez que os nós funcionam com baterias, que a princípio não serão trocadas nem recarregadas.

2. As RSSFs precisam funcionar de forma autônoma sem controle externo.

3. A rede precisa ser resistente a falhas de todos os tipos: nós que deixam de funcionar por falta de energia; sensores de baixo custo que produzem leituras erradas e comunicação via rádio deteriorada por interferência externa.

4. Os dados gerados periodicamente ou orientados a eventos precisam ser levados até nós específicos, que cumprem o papel de interface para coleta de dados. Estes nós recebem a denominação de *sinks* e, em alguns casos, também executam um pós-processamento.

5. O modelo de comunicação em múltiplos saltos é uma imposição face ao alcance reduzido dos transceptores e às distâncias entre os nós e o *sink*.

RUIZ *et al.* (2004) apresentam um sumário de como as RSSF podem ser organizadas para serem melhor estudadas. Os parâmetros de classificação propostos são os seguintes: configuração, sensoriamento, comunicação e processamento. Do referido trabalho, extraiu-se as Tabelas 1.1 e 1.2, que sumarizam a classificação das RSSF conforme sua configuração e modelo de comunicação, respectivamente. Estas tabelas servirão como referência para delimitar o escopo deste trabalho.

Observando a Tabela 1.1, pode-se dizer que, em relação à configuração, os algoritmos propostos neste trabalho consideram RSSF **homogêneas**, de organização **plana** e constituídas por nós sensores **sem mobilidade**. Com relação aos parâmetros **densidade** e **distribuição**, todos os cenários foram considerados.

Em relação ao modelo de comunicação, pode-se dizer que este trabalho considerou RSSF com disseminação dos dados **programada**, por meio de conexões **assimétricas**, utilizando a transmissão de dados no modo ***half-duplex***. A alocação do canal é, com certeza, o parâmetro de classificação com maior relevância para

Configuração		
Composição	Homogênea	Todos os nós da rede têm as mesmas capacidades
	Heterogênea	A rede possui nós com capacidades diferentes
Organização	Hierárquica	Os nós são organizados em grupos (*clusters*). Cada grupo elege um líder (*cluster-head*). Os grupos podem organizar-se hierarquicamente
	Plana	Todos os nós da rede exercem o mesmo papel. Não há formação de grupos nem eleição de líderes
Mobilidade	Estacionária	Os nós da rede não têm mobilidade
	Móvel	Nós com mobilidade
Densidade	Balanceada	Densidade de nós (por unidade de área) ideal
	Concentrada	Elevada densidade de nós
	Esparsa	Baixa densidade de nós
Distribuição	Irregular	Rede com distribuição de nós não uniforme
	Regular	Rede com distribuição de nós uniforme

Tabela 1.1: Classificação das RSSF segundo o parâmetro configuração

este trabalho, uma vez que os algoritmos propostos têm por objetivo suportar, de forma eficiente (baixo consumo de energia), o controle de acesso ao canal por divisão do tempo (TDMA). Em particular, propõem-se soluções para o escalonamento das transmissões e a sincronização de relógios, premissas fundamentais para o funcionamento do referido modelo. Com relação ao fluxo da informação, este trabalho considerou o modelo *Multicast* para estabelecimento do escalonamento de enlaces e para a transmissão de dados sensoreados (aplicação). Em relação ao mecanismo de sincronização de relógios, o modelo adotado foi o *flooding*.

Uma característica comum da maior parte dos cenários das RSSFs é que os nós sensores são desdobrados somente para monitorar o ambiente e levar os dados para um nó específico (o *sink*). Em particular, quando os nós detectam um efeito significativo, não se espera que eles executem qualquer outra ação, pois isso poderia drenar excessivamente a capacidade das baterias.

1.2 Desafios e motivações

O uso eficiente da energia disponível, por parte dos nós sensores, é um fator crítico para as RSSF. Uma vez que o subsistema de comunicação é maior consumidor de energia na composição de um nó sensor, é fundamental que se desenvolvam mecanismos para se evitar as principais fontes de desperdício. Cinco aspectos da comunicação sem fio foram identificados em DEMIRKOL *et al.* (2006): colisões, *overhearing*, *overhead*, *idle listening* e *overmitting*.

- Colisão: ocorre quando um ou mais pacotes de dados chegam ao receptor no momento que este já se encontra no curso da recepção de outro pacote. Tal

Comunicação		
Disseminação	Programada	Os dados são transmitidos em intervalos regulares
	Contínua	Os dados são transmitidos continuamente
	Sob demanda	A disseminação ocorre em função de consultas ou eventos programados
Conexão	Simétrica	Para cada enlace, o comportamento do canal é o mesmo nas duas direções
	Assimétrica	Não há garantia que o canal se comporte da mesma forma nas duas direções de um enlace
Transmissão	Simplex	Nós que transmitem mas não recebem
	Half-duplex	Nós transmitem e recebem em intervalos de tempo distintos
	Full-duplex	Nós podem transmitir e receber simultaneamente
Alocação de canal	Estática	Cada nó da rede tem uma referência fixa para exploração do canal. Não há sobreposição de exploração do canal relativa à referência. TDMA (tempo), FDMA(frequência), CDMA (códigos para espalhamento espectral), SDMA (mesmo canal, enlaces distintos).
	Dinâmica	Ocorre disputa entre os nós para utilização do canal. Quando um nó perde a disputa, entra em contenção, ou seja, aguarda um período aleatório para tentar de novo.
Fluxo da informação	*Flooding*	Os dados sensoreados chegam ao ponto de acesso por meio de seguidos *broadcasts*
	Multicast	Os dados são transmitidos para destinatários previamente selecionados
	Unicast	Os dados transmitidos têm um único destino
	Gossiping	Atualização da informação e envio para destinatários selecionados
	Bargaining	O envio da informação ocorre sob demanda

Tabela 1.2: Classificação das RSSF segundo o modelo de comunicação

inconveniente está relacionado ao problema do terminal escondido, ilustrado na Figura 1.2. A situação que configura o terminal escondido se dá quando os vértices A e C não sabem da presença um do outro, e executam a transmissão para B com alguma sobreposição temporal. Desta forma o terminal B, num dado instante, estará recebendo transmissões simultâneas (colisão), o que forçará o descarte do pacote perdido.

- *overhearing*: acontece quando um nó recebe pacotes não destinados ao próprio. A recepção e o processamento de mensagens não aproveitadas representam um gasto de energia considerável e impactante no tempo de vida da rede.

- *overhead*: pode ser descrito como a relação entre carga de controle e carga

total do protocolo de comunicação. Quanto menor o valor desta fração menor o desperdício de energia.

- *idle listening*: diz respeito ao fato dos rádios permanecerem ligados em períodos ociosos. Este problema está relacionado a uma característica inerente dos transceptores de curto alcance, comumente utilizados nas aplicações das RSSFs. O consumo de potência em períodos ociosos é da mesma ordem de magnitude da potência consumida para transmissão ou recepção de dados (REASON e RABAEY (2004)), conforme observado na Tabela 1.3.

- *overmitting*: ocorre quando o destinatário de uma transmissão ainda não está pronto para realizar a recepção.

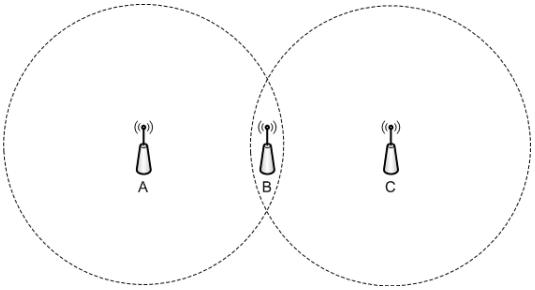

Figura 1.2: O problema do terminal escondido

O fato da comunicação sem fio ser a maior responsável pelo consumo de energia em um típico nó sensor sugere que ganhos consideráveis podem ser realizados na camada de enlace, onde os protocolos de Controle de Acesso ao Meio(MAC) , exercem o controle direto dos transceptores. Protocolos MAC, especificamente desenvolvidos para as RSSFs, apresentam uma clássica relação de compromisso entre vazão, latência e justiça contra a redução no consumo de energia para a maximização do tempo de vida da rede. Cada protocolo MAC tem sua própria política de desligamento dos rádios, em função das transmissões e recepções de cada sensor.

A tabela 1.3, disponibilizada em LANGENDOEN (2008), apresenta os principais parâmetros de desempenho de quatro plataformas de nós sensores, denotando a severa limitação de recursos imposta a esta nova tecnologia. Nota-se que as CPUs vem se tornando cada vez mais rápidas, com maior disponibilidade de memória e com rádios transmitindo em taxas cada vez mais altas. Mais impressionante, é que o consumo manteve-se estável por volta dos 100 mW. O efetivo tempo de vida do nó sensor, no entanto, depende consideravelmente de quanto tempo ele fica dormindo, ou seja, com CPU e rádio desligados. Utilizando os parâmetros da tabela 1.3, e considerando 18.000 joules como a energia equivalente a duas baterias AA, pode-se notar que um nó sensor tem em média 100 horas de vida, trabalhando ligado o

tempo todo. Isto demonstra que a gestão do subsistema rádio (protocolos MAC) é de vital importância para o prolongamento do tempo de vida da rede.

Tabela 1.3: *Especificações de plataformas comerciais para as RSSFs*

	René 1999	**Mica-2 2002**	**Tmote Sky 2005**	**Imote2 2007**
CPU	**ATMEL 8535** 8-bit, 4 MHz 36 μW sleep 60 mW active	**ATmega128L** 8-bit, 8 MHz 36 μW sleep 60 mW active	**TI MSP430** 16-bit, 8 MHz 15 μW sleep 5.4 mW active	**Intel PXA271** 32-bit, 13-416 MHz 390 μW sleep $\geq 31mW active$
Memory	512B RAM 8KB Flash	4KB RAM 128KB Flash	10KB RAM 48KB Flash	32MB RAM 32MB Flash
Radio	**RFM TR1000** 10 Kbps 2 μW sleep 12 mW receive 36 mW xmit 0.5 ms setup	**CC1000** 76 Kbps 100 μW sleep 36 mW receive 75 mW xmit 2 ms setup	**CC2420** 250 Kbps 60 μW sleep 63 mW receive 57 mW xmit 1 ms setup	

Um aspecto complicador está no fato de que os enlaces sem fio são, por natureza, não confiáveis. Desta forma, se não houver retransmissão dos pacotes perdidos, a informação será simplesmente descartada. O grande desafio está em selecionar os melhores instantes para se realizar estas operações. Os protocolos baseados na técnica CSMA (*carrier sense multiple access*) , monitoram o canal para evitar a colisão e usam *backoffs* (intervalos de tempo aleatórios para tentar uma nova transmissão) para resolver o problema de disputa pelo uso do canal. LIU *et al.* (2008) demonstram, em experimentos com plataformas específicas para RSSF, que fontes de interferência suficientemente afastadas produzem cerca de 70% de perda de pacotes, ou seja, apesar da monitoração do canal não indicar motivo para contenção, os dados transmitidos não são corretamente recebidos pelos destinatários (o transceptor não consegue discernir entre informação e interferência). Uma solução imediata seria aumentar a sensibilidade do protocolo de contenção, ou seja, configurá-lo para considerar níveis de potência mais baixos como interferência. Tal medida, no entanto, acarretaria o uso mais frequente dos períodos de contenção e o consequente aumento da latência no encaminhamento das mensagens. Portanto, o simples ajuste do limiar de monitoração do canal não é suficiente para solução do referido problema.

O método mais eficiente para conservação de energia em uma RSSF é o estabelecimento de um ciclo de trabalho, conhecido na literatura como *duty-cycling*. O *duty-cycling* pode ser aplicado em rede, em um nó individual, ou em ambos. Em redes esta abordagem exerce o papel de um controlador de topologia, definindo que subconjunto de nós estarão ativos a cada momento. No que se refere a este trabalho, tal abordagem será aplicada diretamente nos nós sensores, particularmente,

controlando o ciclo de trabalho dos transceptores. Como destacado anteriormente, o ciclo de trabalho de um rádio deve ser definido em função da fração do tempo que o mesmo está no modo de transmissão, no modo de recepção e desligado. O grande desafio é estabelecer, de forma distribuída, um controle local que permita o uso do modelo TDMA, de forma a se evitar a contenção no canal, as colisões de pacotes, o *idle-listening*, o *overhearing* e o *overmitting*.

Protocolos MAC para RSSF, baseados na técnica *duty-cycling* podem ser grosseiramente classificados em síncronos, assíncronos e híbridos. Estas abordagens têm como principal motivação reduzir os períodos em *idle-listening*. Protocolos síncronos como o S-MAC em YE *et al.* (2002) e T-MAC (VAN DAM e LANGENDOEN (2003)) estabelecem um escalonamento que define, individualmente, os períodos de um frame de comunicação destinados ao desligamento dos rádios. Protocolos assíncronos como o B-MAC (POLASTRE *et al.* (2004)) e o WiseMAC (EL-HOIYDI e DECOTIGNIE (2004)) não têm a estrutura de um frame para comunicação, e por isso fazem uso de uma sinalização própria para solicitar a ativação dos rádios dos destinatários, a fim de que se realize a transmissão dos dados.

A vantagem dos protocolos assíncronos está no fato do emissor e receptor estarem completamente desacoplados em seus regimes de trabalho. A simplicidade do projeto elimina a necessidade de sincronização para o correto escalonamento do tempo e o consequente *overhead*. O problema desta abordagem é justamente o alto custo da sinalização, seja pelo aumento do período em que os rádios ficam ativos, seja pela ocorrência de colisões.

Considerando a delimitação do escopo deste trabalho, indicada na seção anterior (1.1.1), pode-se inferir que o grande desafio que se apresenta é o desenvolvimento de algoritmos distribuídos, capazes de prover uma infraestrutura de comunicação TDMA, na qual os aspectos da comunicação sem fio, considerados verdadeiros vilões do desperdício de energia nas RSSF, sejam praticamente eliminados. Inúmeras são as soluções para o estabelecimento de um escalonamento, mas poucas delas apresentam uma abordagem completamente distribuída. Ainda assim, o mecanismo de sincronização, responsável direto pelo bom funcionamento do modelo TDMA, é apontado como grande obstáculo da adoção desta alternativa no âmbito das RSSF, seja pela alta complexidade de uma implementação distribuída, seja pelo alto custo da manutenção da sincronização.

1.3 Objetivos

Este trabalho tem por objetivo investigar e propor algoritmos distribuídos para a implementação de um protocolo TDMA, específico para as RSSF, capaz de evitar o desperdício de energia causado por aspectos inerentes da comunicação sem fio

e, consequentemente, prolongar o tempo de vida destas redes. Para alcançar este objetivo, realizou-se pesquisa sobre mecanismos de escalonamento do tempo e de sincronização de relógios em RSSF. Os trabalhos pesquisados foram analisados à luz dos principais requisitos das RSSF e o produto desta crítica permitiu a proposição de algoritmos distribuídos aderentes a essa tecnologia, ou seja, de baixa complexidade e compensadores em termos de consumo de energia.

A avaliação de desempenho dos referidos algoritmos revelou que os objetivos foram atingidos, se mostrando viáveis para a implementação de protocolos TDMA, livre de colisões. A discussão destes resultados será explicada no Capítulo 5.

1.4 Contribuições

As principais contribuições deste trabalho são as seguintes:

1. O desenvolvimento de dois algoritmos distribuídos e anônimos para o escalonamento de enlaces (transmissões e recepções), de maneira que os nós da rede possam operar no modo TDMA. Estes algoritmos basearam-se na coloração à distância-2 em grafos (nós e arestas), de modo a prover um ambiente de comunicação livre de colisões. A descrição dos algoritmos, bem como os resultados das simulações permitiram a publicação de um artigo científico em conferência internacional especializada (PINHO *et al.* (2009)).

2. O desenvolvimento de um algoritmo distribuído de sincronização de relógios baseado na propriedade gradiente (FAN e LYNCH (2004)). Este novo algoritmo se mostrou bastante simples, apesar de solucionar questões ainda desconsideradas pela comunidade científica. Um novo artigo científico foi publicado em conferência internacional (PINHO *et al.* (2012)).

3. O estudo de viabilidade da integração dos algoritmos citados para a construção de um protocolo MAC TDMA. Os experimentos comprovaram a eficácia da proposta, elevando substancialmente o tempo de vida da rede quando comparada com um esquema tradicional CSMA/CD.

1.5 Organização do Trabalho

Este texto está dividido em seis capítulos. No Capítulo 2 é apresentado um estudo acerca dos principais trabalhos científicos referentes a algoritmos para escalonamento de transmissões e sincronização de relógios, ambos em ambiente de RSSF. No Capítulo 3, são propostos e avaliados dois algoritmos distribuídos e anônimos para o estabelecimento de um esquema de escalonamento de transmissões. Os dois

algoritmos são baseados em soluções para coloração a distância-2 em grafos. O Capítulo 4 apresenta uma proposta de algoritmo distribuído para sincronização de relógios em RSSFs, com base na propriedade gradiente. O Capítulo 5, descreve os detalhes para a implementação de um modelo de comunicação TDMA, utilizando os algoritmos apresentados nos Capítulos 3 e 4. No mesmo capítulo é apresentado um estudo de viabilidade do referido modelo e uma avaliação comparativa entre a abordagem deste trabalho e um protocolo de contenção. Finalmente, no Capítulo 6, são apresentadas as conclusões e os direcionamentos futuros desta pesquisa.

Capítulo 2

Revisão Bibliográfica

As RSSF tornaram-se rapidamente uma área de grande interesse em termos de pesquisa para a indústria e academia. Hoje em dia, o enorme potencial desta tecnologia pode ser facilmente identificado, juntamente com a suas dificuldades inerentes. Basta olhar para o número de projetos de pesquisa sendo financiados nas comunidades europeia e norte-americana, gerando grande quantidade de trabalhos publicados e resultando em produtos a serem comercializados. Estes fatos revelam a clara evidência da crescente importância desta tecnologia. Para ratificar a última afirmação, o Massachusetts Institute of Technology classificou recentemente as RSSFs como uma das dez tecnologias emergentes que irão mudar o mundo, (GARCÍA HERNANDO *et al.* (2009)).

Uma característica marcante desta nova tecnologia é o fato de que após a implantação da rede, os nós sensores funcionarão até suas fontes de energia se esgotarem. Depois disso esses dispositivos são simplesmente descartados. Cabe ressaltar que as fontes de energia disponíveis para esta tecnologia têm capacidade bastante limitada. Sendo assim, o tempo de vida dessas redes é fator decisivo para considerar sua viabilidade.

Uma vez que o rádio é considerado o principal consumidor de energia em um nó sensor, torna-se imperativo uma política bastante cuidadosa para regular o seu uso. Aliado a isto, pode-se dizer que o modelo de tráfego em aplicações desta nova tecnologia é fortemente correlacionado no tempo e no espaço, em função da proximidade de vários nós aos eventos monitorados. A principal consequência deste modelo é o alto índice de colisões, efeito indesejado quando se pretende economizar energia. Desta forma, um dos grandes desafios para pesquisadores e engenheiros é o desenvolvimento de protocolos de acesso ao meio que otimizem a utilização dos transceptores para permitir a máxima economia de energia. Pode-se afirmar ainda que as limitações impostas pela simplicidade típica de hardware, embarcada nos sensores implicam soluções que executem o mínimo de processamento e que minimizem o uso de memória. Estas considerações limitam as fronteiras do desenvolvimento

dos protocolos MAC.

Este capítulo tem por objetivo apresentar um estudo acerca dos principais e mais recentes trabalhos científicos relacionados a algoritmos para escalonamento de transmissões e algoritmos de sincronização de relógios para as RSSF.

2.1 Algoritmos para Acesso ao Canal em RSSF

Este trabalho considera a classificação usada em LANGENDOEN (2008), distinguindo três classes de acesso ao meio.

2.1.1 Acesso Aleatório

Nesta abordagem os nós não tem qualquer organização no domínio do tempo e disputam o canal rádio de acordo com a sua própria demanda. Em resumo, os nós monitoram a atividade do canal antes da transmissão. Em particular, se o canal estiver ocupado, o nó que pretende transmitir entra em contenção, ou seja, aguarda um período aleatório (*backoff*) para tentar transmitir novamente. Esta técnica é conhecida pela sigla CSMA, do inglês, ***C****arrier* ***S****ense* ***M****ultiple* ***A****ccess*.

Esta abordagem propicia uma grande flexibilidade para lidar com redes de diferentes densidades e cargas de tráfego, de maneira que nenhuma ação específica precisa ser executada antes da implantação da rede. As mudanças (por exemplo, um nó que entra na rede) são facilmente absorvidas. Outra vantagem vem do fato de que os nós não precisam sincronizar seus relógios. O lado desfavorável é que boa parte da energia disponível é desperdiçada em função das colisões e do problema de *idle listening*.

Um esquema bastante interessante para lidar com o problema de *idle listening* é o *Low-Power Listening and Preamble Sampling* (HILL e CULLER (2002)). A ideia é que cada mensagem seja precedida por um preâmbulo de modo a alertar potenciais receptores sobre a iminência da transmissão de dados. Os nós que perceberem este preâmbulo, manterão seus rádios ligados até o final da recepção da mensagem. A garantia do êxito desta abordagem é alcançada com o ajuste do preâmbulo, de tal maneira que este tenha duração superior ao período de desligamento do rádio (*sleep*), ou seja, o receptor sempre será acordado para a recepção da mensagem. Uma maneira conveniente de implementar esta estratégia é estender a duração do preâmbulo padrão (parte do cabeçalho da camada física). A Figura 2.1 ilustra este procedimento de amostragem periódica. Uma vez que a quantidade de receptores é bem superior ao número de transmissores e o tráfego de dados é bastante reduzido, a economia de energia torna-se relevante. Um obstáculo em potencial para este tipo de abordagem está no fato que tanto a recepção quanto o *overhearing* tornam-se bem

mais custosos pois o período entre o momento que o receptor percebe o preâmbulo (acorda) e aquele em que recebe o início da mensagem é, na média, a metade do período do preâmbulo.

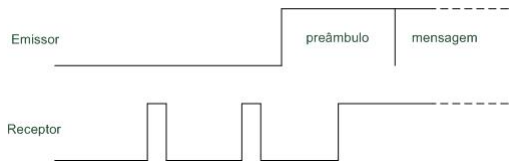

Figura 2.1: O preâmbulo extendido permite uma verificação eficiente do canal

Nota-se que para qualquer cenário (vazão de dados, densidade de nós) existe um período ótimo para os nós ficarem acordados. O protocolo B-MAC (POLASTRE *et al.* (2004)) permite a configuração em tempo real desse período, de maneira a possibilitar a otimização da economia de energia, em função da mudança de comportamento da rede. Outra contribuição do B-MAC está no procedimento monitoração do canal. Ao invés de executar uma única amostragem, o B-MAC realiza cinco amostragens consecutivas e avalia que o canal está livre se uma das medidas ficar abaixo de um determinado nível de referência. Este procedimento elimina de forma eficiente o ruído e a interferência, principais causadores de alarmes falsos de canal ocupado.

WiseMAC (EL-HOIYDI e DECOTIGNIE (2004)) foi o primeiro protocolo que implementou uma melhoria na estratégia de se amostrar o preâmbulo das mensagens. A ideia é que os transmissores adquiram conhecimento do período em que seus vizinhos acordam para verificar o canal. Essas informações seguem anexadas aos *acknowledges* das mensagens. A partir daí, escalona-se a transmissão das mensagens o mais próximo possível do momento em que seus vizinhos vão acordar para avaliar o canal. Este procedimento permite a substancial redução da duração do preâmbulo, um grande causador de interferência, contribuindo para a economia de energia da rede.

Uma proposta diferente para redução dos danosos efeitos causados pelo problema do *idle listening* denomina-se Wake-up Radio (LANGENDOEN (2008)). Nesta abordagem, os nós são equipados com um segundo rádio, cujo consumo é extremamente baixo. Sua utilização é exclusiva para sinalização de ocupação de canal. Desta forma, o rádio primário só é ligado quando existir a confirmação de que novas mensagens serão enviadas. O problema associado a esta proposta está no fato que os rádio secundários têm uma sensibilidade muito elevada e a sinalização é transmitida em *broadcast*. Neste cenário, grande parte dos nós serão acordados sem necessidade.

2.1.2 Acesso baseado em ciclo de trabalho

Nesta abordagem, também conhecida na literatura como *Slotted access*, o tempo é dividido em *slots* e os nós precisam estar sincronizados segundo uma referência de tempo de forma que possam ser ligados coletivamente, no início de cada *slot*, e desligados imediatamente depois que a troca de mensagens estiver concluída. O S-MAC (YE *et al.* (2002)) é o primeiro exemplo desta classe, aperfeiçoando o CSMA/CA, com a implementação de um ciclo de trabalho fixo e uma política para se evitar *overhearing*. A parte complicada deste protocolo está justamente na sincronização dos nós. Isto é feito de maneira que cada nó, regularmente, transmite, em *broadcast*, pacotes de sincronismo, incluindo aí, a sua própria base de tempo, o que permite a outros nós compensar os desvios de seus relógios.

A simplicidade do S-MAC (ciclo de trabalho fixo) tem dois grandes obstáculos: o primeiro é que a responsabilidade de se ajustar um ciclo de trabalho ótimo, antes mesmo da implantação da rede, fica a cargo do desenvolvedor. O segundo está no fato que as flutuações de tráfego só podem ser resolvidas com a maximização da fração do ciclo de trabalho em que os nós fiacm ligados, o que definitivamente vai de encontro a ideia do rádio só estar ligado quando for necessário. Para tratar destas questões, o protocolo Timeout MAC (T-MAC) (VAN DAM e LANGENDOEN (2003)) introduziu o período adaptativo de ativação. Os nós só escutam o meio por um curto período (15 ms para o T-MAC contra 300 ms para o S-MAC), no início de um *slot* e, caso não haja comunicação em curso, voltam a dormir. Por outro lado, se o nó continuar engajado na comunicação, ele pode escalonar um novo momento para escutar o meio, imediatamente após a última transmissão. Simulações mostraram que o T-MAC é capaz de lidar com flutuações de tráfego no tempo (baseada em eventos) e no espaço, superando o S-MAC na economia de energia por um fator de 5.

O protocolo denominado Schedule Channel Polling MAC (SPC-MAC), de YE *et al.* (2006), otimiza o modelo de contenção. Neste protocolo, cada *slot* começa com uma janela de contenção. O nó que tem a intenção de transmitir alguma mensagem escolhe um momento aleatório, dentro desta janela, para ligar o seu rádio, verificar o canal e iniciar a transmissão de um preâmbulo, caso o canal esteja livre. O preâmbulo dura até o final da janela de contenção, de forma a bloquear a ação de outros potenciais emissores. Se não houver tráfego, o SPC-MAC realiza apenas uma verificação do canal por *slot*, tornando-o o mais eficiente desta classe.

Todos os protocolos apresentados nesta seção priorizaram a eficiência energética sem qualquer preocupação com o problema da latência. Com o S-MAC a comunicação em múltiplos saltos depende diretamente da largura da janela de ativação. Com o T-MAC, o número de saltos é limitado a três. Já com o SCP-MAC, somente

uma mensagem pode ser transmitida por *slot*. O protocolo Data gathering MAC (D-MAC) (LU *et al.* (2004)) aborda esta questão especificamente para o padrão de comunicação *convergecast*, no qual é presumida uma estrutura que define uma topologia em árvore para escoamento dos dados até um *sink* . A ideia básica é defasar os tempos de ativação, de acordo com o nível do nó na árvore, de modo que os dados possam fluir das folhas para a raiz da maneira mais rápida possível. A Figura 2.2 ilustra tal procedimento. Os nós escutam seus filhos e em seguida retransmitem as mensagens para os seus pais. Os nós que perderam na contenção não precisam aguardar um novo fluxo de subida, mas sim tentar enviar a mensagem em outro *slot*. Esta estratégia aumenta a capacidade do D-MAC, tornando-o, ainda, adaptativo às flutuações de tráfego. O ponto fraco deste protocolo está no fato de não ser flexível para suportar outros padrões de comunicação.

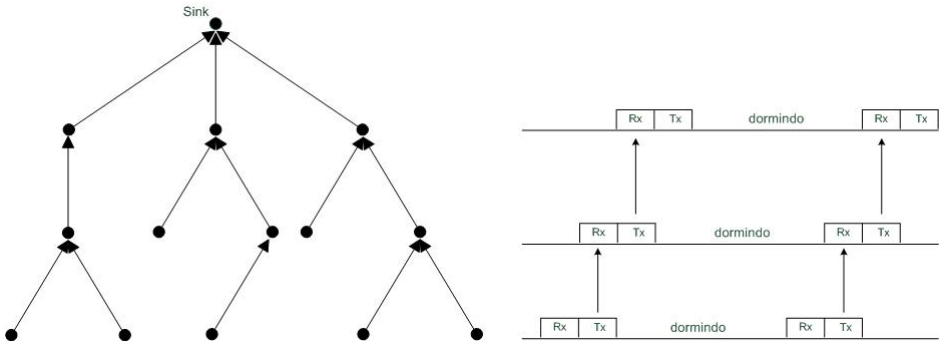

Figura 2.2: DMAC: o nível dos nós na árvores define a defasagem do acesso ao canal

2.1.3 Acesso baseado em quadros (*Frame-based access*)

A terceira classe de protocolos MAC agrupa os *slots* em quadros (*frames*), implementando-se um escalonamento no qual os pares enlace-*slot* são definidos segundo alguma regra de construção da rede. A grande vantagem desta abordagem está no fato de que implementações bem planejadas eliminam a possibilidade de haver colisões e reduzem drasticamente os problemas de *idle listening* e de *overhearing*. Quando os enlaces de comunicação são escalonados, ou seja, define-se para cada *slot* um par transmissor-receptor, os receptores tem a capacidade de ligar seus rádios, somente nos momentos em que houver uma transmissão planejada, eliminando todo o *overhearing*. No caso de escalonamento apenas dos transmissores, os nós precisam escutar todos os *slots*, mas podem evitar boa parte do *overhearing* desligando os seus rádios logo depois de verificar no cabeçalho MAC que a mensagem não era endereçada aos mesmos. Em ambas as variantes, o problema de *idle-listening* pode ser reduzido a uma simples verificação de utilização do canal. Aproveitando esta potencialidade, vários protocolos MAC que foram desenvolvidos com base em um TDMA

clássico, fazendo uso de um ponto de acesso (estação central), tiveram que ser adequadamente modificados para operarem desassistidos de qualquer infra-estrutura nas RSSFs.

A primeira abordagem para a implementação de um TDMA distribuído para as RSSFs foi o desenvolvimento do protocolo **Power Efficient and Delay Aware Medium ACcess - PEDAMACS** (ERGEN e VARAIYA (2006)). Neste trabalho assume-se que o nó *sink* está aparelhado com um rádio de alta potência que possibilita a cobertura de todos os nós da rede. Isto permite ao *sink* sincronizar os nós e escalonar transmissões e recepções. Uma vez que muitos nós não tem cobertura rádio direta para o sink, e por isso não podem informá-lo sobre suas demandas, o PEDAMACS inclui um procedimento especial de inicialização. Primeiramente, o *sink* constrói uma árvore de conexões, difundindo um pacote específico contendo uma variável (número de saltos) com o valor unitário. Os nós que receberem este pacote incrementam a variável e retransmitem o pacote. Este procedimento se repete até que toda a rede seja alcançada. Isto permite que todos os nós da rede tenham conhecimento de sua topologia local. Em seguida, todos os nós transmitem sua topologia local (pais e filhos) e suas demandas de transmissão de dados (este procedimento é realizado periodicamente) na direção do *sink*. Depois que a informação de todos os nós chegar ao *sink*, este terá completo conhecimento da topologia da rede e poderá estabelecer um escalonamento global, livre de colisões, que será transmitido em *broadcast* para toda a rede. A partir de então, os nós iniciam a fase de coleta de dados, recebendo e transmitindo mensagens, conforme o escalonamento implementado. Para lidar com mudanças de topologias ocasionais, devido a mobilidade dos nós ou interferência externa, o PEDAMACS realiza, ocasionalmente, um procedimento de ajuste, no qual os nós reportam as alterações observadas.

A premissa de que o *sink* tem cobertura rádio para toda a rede é questionável por conta dos obstáculos que podem bloquear a linha de visada entre os nós, e também pelos efeitos nocivos da propagação em múltiplos caminhos (*multi-path*). Uma segunda questão a ser considerada para a análise de viabilidade deste protocolo, diz respeito ao uso da técnica CSMA no procedimento de inicialização. Esta abordagem não elimina completamente as colisões, e em função disso, alguns nós podem ficar fora da árvore de conexões, o que determina seu completo silenciamento na fase de coleta de dados.

RAJENDRAN *et al.* (2006) assumiram que todos os nós são idênticos e que o escalonamento será definido por uma função distribuída. No protocolo desenvolvido por eles, **TRaffic-Adaptative Medium Accesss (TRAMA)**, os nós transmitem regularmente, em *broadcast*, informações sobre seus fluxos de dados, assim como a identidade de seus vizinhos. Todo nó toma conhecimento de todos os nós na sua vizinhança de dois saltos. Com tal informação, obter-se-á um escalonamento livre

de colisões por meio de uma função *hash* distribuída que determina um transmissor para cada *slot*. A informação sobre o fluxo de tráfego dos vizinhos mais próximos (vizinhança de 1 salto) é usada para resolver os empates em favor dos nós mais congestionados. Para reduzir o *overhearing*, todo transmissor inclui uma lista em cada pacote transmitido, que detalha os próximos receptores para os quais planeja transmitir, nos próximos 100 *slots*. Para se ajustar a diminuição de tráfego, um nó simplesmente exclui do seu planejamento (lista) as previsões de transmissões, que no momento são consideradas desnecessárias para atender a demanda do referido nó. Isto permite que outros nós assumam capacidades limitadas para se adaptar a novas demandas, promovendo um ajuste distribuído do fluxo de dados em toda a rede.

Cabe ressaltar que para o perfeito funcionamento do mecanismo de escalonamento, a informação sobre o fluxo de tráfego precisa ser transmitida em *broadcast* a cada 100 *slots*, o que provoca considerável *overhead*. Em função disso, buscou-se o aprimoramento do TRAMA com o desenvolvimento de um novo protocolo, denominado **FLow-Aware Medium Access (FLAMA)** (RAJENDRAN *et al.* (2005)). A principal mudança do funcionamento deste protocolo, diz respeito ao mecanismo de transmissão das informações sobre fluxo de dados. Ao invés de transmiti-las a cada 100 *slots*, os nós só o fazem, quando receberem requisições explícitas para tal. Apesar desta estratégia reduzir consideravelmente o *overhead*, a redução no consumo de energia não foi significativa. O escalonamento global do protocolo FLAMA inclui alguns *slots* especiais para inicializar e acomodar novos nós que venham aderir a rede.

O protocolo **Lightweight MAC** (LMAC) (VAN HOESEL e HAVINGA (2004)), criado por van Hoesel et al., também faz uso de um mecanismo distribuído de seleção de *slots*, baseado na informação da vizinhança de dois saltos, No entanto, diferentemente do TRAMA e do FLAMA, o escalonamento não depende do número do *slot*, tornando-o mais trivial na organização da rede. Cada nó possui um *slot* de tamanho fixo, no qual ele sempre transmite um cabeçalho e logo a seguir, opcionalmente, os dados (*payload*). O cabeçalho contém vários campos, entre os quais, o destino e o tamanho do *payload*. Diferente de muitos protocolos MAC, o correto recebimento das mensagens não é confirmado para se diminuir o consumo de potência com tal transmissão. O LMAC deixa a questão da confiabilidade dos enlaces para as camadas superiores.

Para facilitar a inclusão de novos nós na rede, cada cabeçalho tem um campo destinado a um conjunto de bits (*bitset*) que detalha quais são os *slots* que estão ocupados na vizinhança de um salto do nó emissor. Executando um OR lógico com todos os *bitsets* recebidos, um novo nó pode facilmente determinar quais *slots* ainda estão disponíveis na vizinhança de 2 saltos.

O mecanismo de seleção de *slots* descrito acima tem um grande inconveniente: o número de nós da vizinhança de 2 saltos não pode exceder o número de *slots* em um quadro, o qual é fixado antes da implantação da rede. Optar por agregar um grande número de *slots* por quadro, implicaria em um super-dimensionamento (em boa parte do tempo muitos *slots* seriam desperdiçados) e em adicional *overhead*, devido aos grandes *bitsets*; se a opção for a contrária, muitos nós podem simplesmente não aderir a rede por falta de *slots* disponíveis. Este problema foi parcialmente resolvido com as melhorias introduzidas no protocolo **Adaptative Information-centric LMAC (AI-LMAC)** de CHATTERJEA *et al.* (2004), permitindo-se que os nós reivindiquem múltiplos *slots* dentro de um mesmo quadro. Esta estratégia também permite aumentar a vazão de dados no padrão de comunicação *convergecast*, alocando-se mais *slots* para os nós mais próximos do *sink*. Em termos de confiabilidade, a decisão de quantos *slots* um nó em particular pode reivindicar é deixada para as camadas superiores.

O trabalho de SHI e FAPOJUWO (2010) propõe uma abordagem para se determinar o escalonamento TDMA de uma RSSF com alta eficiência energética e latência mínima. Este procedimento é dividido em dois passos distintos. No primeiro passo, emprega-se uma modelagem de otimização *cross-layer*, com o objetivo de se definir o custo de cada enlace, e consequentemente, os fluxos de dados com maior eficiência energética. De posse deste mapeamento, um único nó *sink* executa do algoritmo *Minimum Delay Scheduling Algorithm* (MDS), que em resumo, analisa os fluxos de maior eficiência e considera o reuso de *slots* associados a nós que não estão nestas rotas. O resultado é diminuição do frame TDMA, diminuindo a latência das mensagens e otimizando o uso dos nós da rede, com respeito ao consumo de energia. Os resultados experimentais confirmam a relevante redução do consumo de energia, no entanto, a alta complexidade e o alto custo para manutenção do escalonamento em função da evolução das mudanças não programadas da rede, ainda são grandes obstáculos para a adoção deste trabalho em boa parte das aplicações de RSSF.

O protocolo **Zebra MAC (Z-MAC)** (RHEE *et al.* (2005)) inicia com um algoritmo distribuído de associação de *slots*, que contabiliza a vizinhança de 2 saltos para estabelecer um escalonamento livre de colisões. Terminada esta etapa, os nós que precisarem transmitir mensagens realizam a contenção para acessar ao canal. Um nó pode realizar a contenção para qualquer *slot*, no entanto terá prioridade no *slot* a ele associado. Isto acontece, em função da janela de contenção ser dividida em duas partes: a inicial, reservada ao nó associado ao slot e a final, reservada para qualquer nó. A Figura 2.3 ilustra este procedimento. Quando um nó verifica que está perdendo muitos pacotes, ele sinaliza em *broadcast* uma notificação para que seus vizinhos realizem a contenção daquele *slot* na parte alta da janela de contenção. Deste modo, a vizinhança de dois saltos fica bloqueada de realizar a contenção para

uso do *slot*, associado a um referido nó, prevenindo-se o efeito do terminal escondido (colisões). Depois de 10 s de *timeout*, os nós voltam a operação normal. Essencialmente, o Z-MAC constrói uma sobre-camada TDMA em cima do CSMA básico, utilizando o B-MAC (*low-power listening*).

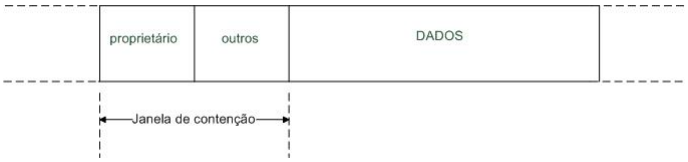

Figura 2.3: Z-MAC: privilégio de acesso aos proprietários do slot

Os protocolos **Pattern MAC (PMAC)** (ZHENG *et al.* (2005)) e **Crankshaft** (HALKES e LANGENDOEN (2007)) compartilham a ideia de escalonamento para a recepção. A vantagem está no fato de que um nó só precisa acordar (ligar o rádio) para verificar a existência tráfego a ele destinado no seu *slot* associado, ao invés de fazê-lo em todos os *slots*, como adotado nos esquemas clássicos, baseados no escalonamento de transmissão. O protocolo Crankshaft realiza a alocação de *slots* com base no identificador (ID) dos nós. O PMAC usa um esquema mais elaborado que leva em consideração a demanda de tráfego de um nó e de seus vizinhos e inclui um campo especial no final de cada quadro, onde cada nó anuncia seu padrão de funcionamento para o próximo quadro. Este padrão, na realidade, é o ciclo de trabalho pretendido, ou seja, para cada *slot* acordado, em quantos *slots* (n) o nó permanecerá dormindo. Isto reduz o estado de um nó ao par (ID,n). A consequência do escalonamento de receptores é que vizinhos podem compartilhar o mesmo *slot*, forçando os emissores a realizar contenção em cada *slot*. Crankshaft usa o eficiente esquema de contenção do SCP-MAC, enquanto o PMAC é baseado no CSMA/CA. Crankshaft tem um desempenho melhor em relação ao consumo de energia. O PMAC é mais adequado quando a aplicação é mais sensível as flutuações de tráfego. O protocolo Crankshaft, no entanto, inclui uma otimização para aumentar o fluxo de dados na direção do *sink*, permitindo ao mesmo receber em todos os *slots*. Simulações mostram que o protocolo Crankshaft supera o SCP-MAC, especialmente em cenários mais densos.

KULKARNI e ARUMUGAM (2006) propõem um algoritmo determinístico e auto-estabilizado para controle de acesso ao meio em sistemas TDMA para RSSFs no qual cada sensor tem conhecimento apenas de sua vizinhança. Este algoritmo baseia-se em uma atribuição inicial de períodos para comunicação (*slots*) que são dinamicamente atualizados por meio de difusão de mensagens enviadas por uma estação base.

ERGEN e VARAJA (2005) apresentam um algoritmo distribuído baseado na coloração de vértices de um grafo de conexões. Este trabalho considera que os

nós sensores utilizam um único ponto de acesso (PA) para a transferência de suas mensagens. Neste trabalho os autores obtiveram um limite superior para estes escalonamentos como função do número total de pacotes gerados na rede.

PANTAZIS *et al.* (2009) propõem um esquema de escalonamento TDMA que faz o balanceamento entre a economia de energia e o retardo da rede. Na referida abordagem, cada nó sensor envia informações sobre roteamento e sobre sua vizinhança para uma estação central que configura o escalonamento. As mensagens de controle são enviadas para os nós sensores por meio de técnicas de *flooding* (inundação).

GANDHAM *et al.* (2008) propõem um algoritmo distribuído para obtenção de um protocolo MAC TDMA baseado no problema de coloração de arestas. O algoritmo é dividido em duas fases. Na primeira fase o algoritmo executa a coloração de arestas à distancia-1, ou seja, nenhum par de arestas incidente em um nó da rede pode ter a mesma cor associada. Em seguida, o algoritmo atribui orientações aos enlaces de maneira a se estabelecer um escalonamento livre de colisões. Se isto não for possível, novas cores são associadas a determinados enlaces, até que o objetivo seja alcançado. O referido trabalho prova que a reversão das arestas leva a uma nova configuração que também tem um escalonamento livre de colisões. O principal resultado deste trabalho é a minimização do número de cores utilizadas que é limitada a $\delta + 1$ cores, onde δ é o grau da rede. Nas duas fases é necessário que os nós tenham identificadores globais. Os resultados desse trabalho foram comparados aos resultados obtidos pelos algoritmos propostos no Capítulo 3.

No próximo capítulo serão apresentados dois novos algoritmos que servirão de base para a implementação de um protocolo MAC livre de colisões. Foram obtidos resultados analíticos para convergência temporal de pior caso. Ainda, por meio de simulações, foram avaliados o desempenho dos algoritmos com relação a algumas métricas. Os resultados alcançados indicaram uma relação de compromisso entre a convergência de tempo e de mensagens contra o número de cores utilizadas.

2.2 Algoritmos para sincronização de relógios em RSSF

O problema de sincronização de relógios em RSSF tem sido estudado intensamente nas últimas duas décadas e ainda não há uma abordagem que seja independente da topologia e da aplicação, capaz de prover elevado grau de precisão na sincronização e lidar com a escalabilidade da rede, (RANGANATHAN e NYGARD (2010)). Talvez esta afirmação seja muito forte, mas, de qualquer maneira, reflete a complexidade do desafio e abre espaço para novas ideias voltadas para o desenvolvimento de algoritmos distribuídos que fazem uso da natureza colaborativa dos nós sensores para

sincronização dos relógios. A sincronização de relógios é um problema importante a ser resolvido pois muitas aplicações dependem deste serviço, tais como: o monitoramento ambiental, o acompanhamento de alvos móveis, a fusão de dados e os esquemas TDMA para escalonamento de enlaces. Considera-se, por exemplo, a aplicação de acompanhamento de alvos móveis, na qual uma rede de sensores é desdobrada em uma área de interesse para a monitoração de objetos que passem naquele local. Quando ocorre um evento, ou seja, um alvo é detectado, os nós sensibilizados registram a localização e o tempo da detecção. Mais tarde, estes dados são encaminhados pela rede e são objeto da agregação (muitas vezes referenciada como fusão), na qual um determinado nó estima a trajetória do objeto detectado. Sem um preciso esquema de sincronização, a trajetória estimada pode ser bem diferente da verdadeira trajetória percorrida pelo objeto em questão.

Existem três razões para que os relógios dos nós sensores apresentem valores diferentes:

(1) Cada relógio inicia seu funcionamento em um tempo universal distinto;

(2) os osciladores a cristais não são perfeitos, ou seja, as frequências de oscilação são suavemente diferentes entre nós distintos. Isto faz com que os valores dos relógios divirjam gradualmente com o passar do tempo. Essa imprecisão é conhecida como *skew error*;

(3) além de variar entre si (*skew*), a frequência dos relógios pode variar com tempo, em função do envelhecimento ou de condições ambientais, tais como a variação de temperatura. Esta imprecisão é conhecida como *drift error*.

A sincronização em redes de sensores sem fio pode ser alcançada de inúmeras maneiras, dependendo dos requisitos da aplicação, da disponibilidade de alguns dispositivos, e recursos como energia e memória. Em geral, a classificação para as diferentes abordagens é representada por pares de características contrárias (ROMER *et al.* (2005)):

- Sincronização pró-ativa ou reativa. A sincronização pró-ativa é realizada de forma contínua a fim de manter toda a rede sincronizada. Um exemplo ocorre quando um nó de referência informa, periodicamente, o valor do seu relógio para os outros nós, permitindo a estes uma estimativa segura do defasamento em relação à referência. Ao contrário, a sincronização reativa ocorre apenas em função de uma demanda específica. Esta situação pode ser verificada quando nós sensores registram os tempos dos eventos e ao enviá-los para um concentrador, inserem no corpo da mensagem o valor do relógio atualizado. Após a coleta de dados, o concentrador executa algum procedimento de sincronização para converter o valor dos relógios dos vizinhos no tempo local. Este

é um exemplo do que é chamado sincronização *post-facto*, (ELSON e ESTRIN (2001)).

- Sincronização com ou sem referencia externa. O uso do GPS é um bom exemplo referência externa, no qual nós da rede são capazes de converter seus relógios para os valores de referência e vice-versa, alcançando assim a sincronização especificada. Na segunda abordagem, a rede toma como referência o relógio de um ou mais nós, escolhidos segundo um critério específico.

- Sincronização global ou local. Na sincronização global, toda a rede mantém uma atividade de sincronização, independente dos eventos a serem monitorados. Na sincronização local, somente os nós sensores, que estiverem envolvidos em atividades que necessitem da sincronização dos relógios, manterão os procedimentos de sincronismo.

- A garantia da precisão pode ser determinística ou probabilística. No primeiro caso o consumo de energia é elevado mas justificável em aplicações nas quais a temporização é fator determinante na garantia de segurança. Alternativamente, pode-se implementar mecanismos que executem a sincronização desejada de forma probabilística. Esta abordagem permite uma maior economia de energia.

Os primeiros esforços para sincronização de relógios em RSSF utilizaram equipamentos GPS como referência global do tempo. Esta abordagem não foi adiante, uma vez que restringia-se a aplicações ao ar livre e com fontes contínuas de energia, situações bem diferentes das encontradas em grande parte dos cenários de atuação das RSSF.

O algoritmo *Fine-grained network time synchronization using reference brodcasts* (ELSON *et al.* (2002)) introduziu a ideia de sincronização entre receptores (*Reference Broadcast Scheme - RBS*). Um nó de referência envia pacotes de sincronização. Estes pacotes não tem qualquer dado a ser utilizado no procedimento de sincronização. A importância destas mensagens está no fato dos receptores registrarem o tempo de recepção nos seus próprios relógios. Após o registro os nós trocam mensagens entre si, cujo conteúdo é o tempo registrado na chegada do pacote de referência. De posse desses valores, cada nó pode estimar qual a sua defasagem em relação a cada vizinho. A Figura 2.4 ilustra bem este procedimento.

Diferente do esquema anterior, o algoritmo *Time-sync protocol for Sensor Networks* (TPSN) (GANERIWAL *et al.* (2003)) usa o modelo de sincronização emissor-receptor. O funcionamento do TPSN é baseado no estabelecimento da topologia de uma árvore. O algoritmo se divide em duas fases: descoberta e sincronização. Na fase de descoberta a rede é organizada de forma hierárquica, associando-

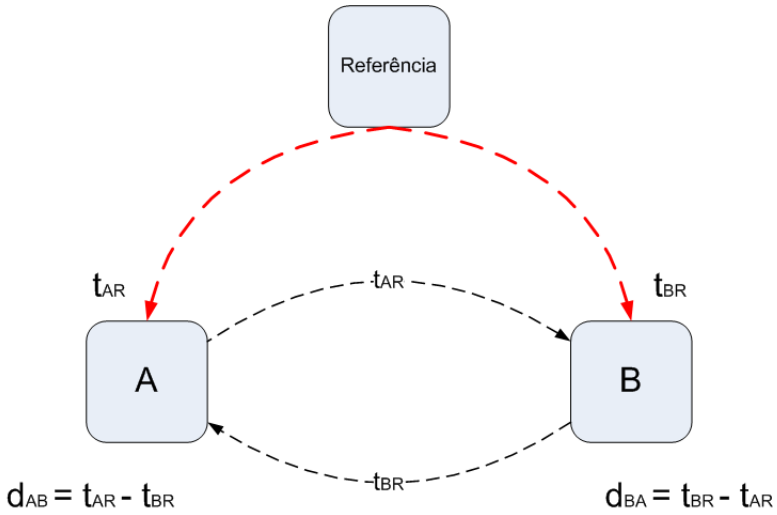

Figura 2.4: *Reference broadcasting*

se a cada nó um nível da hierarquia. Sendo assim, apenas a um nó da rede associa-se o nível zero, o nó raiz. Na fase de sincronização todos os nós do nível i se sincronizarão com os seus respectivos pais na árvore, que estão no nível $i - 1$. Este procedimento fará com que todos os nós da rede fiquem sincronizados com o nó raiz.

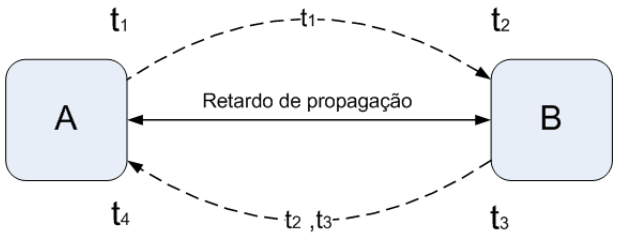

Figura 2.5: Modelo de sincronização emissor-receptor

O conceito da sincronização emissor-receptor pode ser entendido com o auxílio da Figura 2.5. O nó A envia um pacote para o nó B contendo o registro do tempo da emissão t_1. Ao receber o pacote, o nó B registra o tempo da recepção t_2. Este valor é igual a soma de t_1, o retardo da propagação do pacote P e *offset* entre os relógios o:

$$t_2 = t_1 + P + o \tag{2.1}$$

Em seguida o nó B envia um pacote para o nó A, contendo t_2 e o registro do tempo da emissão t_3. Finalmente o nó A, ao receber o segundo pacote, registra o tempo da recepção t_4. Assumindo uma condição simétrica para a propagação, tem-se que:

$$t_4 = t_3 + P - o \tag{2.2}$$

Subtraindo-se 2.2 de 2.1 chega-se ao valor do *offset* entre os relógios de A e B:

$$t_2 - t_4 = t_3 + P - o - t_1 - P - o \tag{2.3}$$

$$o = \frac{t_3 - t_1 - t_4 + t_2}{2} \tag{2.4}$$

Desta forma, após a troca de mensagens, o nó A pode estimar o valor do relógio do nó B. Da mesma maneira que na fase de descoberta, a fase de sincronização inicia-se com o nó raiz e se propaga por toda a rede. A grande fonte de incerteza deste procedimento reside na tomada dos tempos da transmissão e recepção dos pacotes, que se forem efetuados na camada de aplicação serão contaminados por retardos aleatórios provenientes do caminho percorrido na pilha de protocolos. Para diminuir esta incerteza no TPSN foi adotada a estratégia de registrar os tempos diretamente na camada MAC. Esta abordagem permitiu ao TPSN apresentar um erro médio de sincronização duas vezes menor que o erro apresentado com técnicas RBS (RANGANATHAN e NYGARD (2010)). O ponto fraco deste algoritmo está no fato de não considerar que os relógios possam divergir, em função do *skew* entre eles, tornando mandatório os frequentes procedimentos de sincronização.

O algoritmo *Flooding-Time Synchronization Protocol* (FTSP) (MARÓTI *et al.* (2004)) implementou algumas melhorias em relação ao TPSN. De forma similar a rede se organiza em uma estrutura com um nó raiz e todos os nós se sincronizam com ele. No entanto, adotou-se a topologia em malha, bem mais versátil que a árvore usada no TPSN. Esta abordagem elimina a fase inicial de descoberta e é mais robusta contra a falha de nós, de enlaces de comunicação e mudanças da topologia da rede.

O nó raiz, único, pode ser reeleito dinamicamente. Este mecanismo de eleição é bem simples e consiste na auto-eleição, caso um nó sensor não receba mensagens de sincronismo por um período pré-determinado. É bem provável que este procedimento abra a possibilidade para múltiplos nós raiz. Para solução desse problema, as mensagens de sincronismo carregam o identificador do nó raiz, que é comparado com o identificador local a cada recepção. Quando um nó raiz receber uma mensagem de sincronismo com identificador maior, o nó local desiste de sua eleição.

O nó raiz dissemina periodicamente a base de tempo global para a rede. Para compensar o *skew* entre o nó raiz e seus descendentes, cada nó faz uma regressão linear para converter o tempo local no tempo do nó de referência. Esta é a primeira abordagem que trata da correção da taxa de progressão dos relógios.

O algoritmo FTSP, no entanto, apresenta um aumento exponencial do erro em função do tamanho da rede. Uma alternativa para superar este problema, foi proposta com o algoritmo PulseSync (LENZEN *et al.* (2009)). Neste algoritmo, a referência de tempo é retransmitida tão rapidamente quanto possível através da rede, permitindo a adaptabilidade às mudanças na topologia da rede e às variações da

taxa de progressão do relógio de referência. Os experimentos foram realizados com 20 nós da plataforma Mica2, dispostos em linha, e revelaram um erro médio global bem inferior ao apresentado pelo FTSP.

O protocolo *Scalable Light Weight Time Sync Protocol* (SLTP) usou uma técnica de clusterização passiva para a criação de *clusters* com o menor custo possível para a rede, além de implementar procedimentos de regressão linear para estimar o valor do relógio de referência. Esta abordagem teve por objetivo principal a redução do consumo de energia. O SLTP divide-se em duas fases: configuração e sincronização. Na fase de configuração, estabelecem-se os *clusters*, cada um com seu líder. Na fase de sincronização os líderes transmitem a sua base de tempo local para os integrantes de seu *cluster*, que por meio do método de regressão linear estimam taxa e offset de seus líderes. Resultados de simulações revelaram uma melhoria de desempenho considerável em relação aos aspectos de consumo de energia, acurácia e escalabilidade, quando comparados com algoritmos similares (LENZEN *et al.* (2009); GELYAN *et al.* (2007)).

FAN e LYNCH (2004) introduziram o problema da sincronização de relógios com propriedade gradiente. Os autores consideraram que um algoritmo de sincronização de relógios usa a propriedade gradiente quando o erro entre relógios de dois nós é limitado pela distância entre eles. Eles chegaram a prova formal de um limite inferior para este erro de $\Omega(d + \frac{\log D}{\log \log D})$, para dois nós a distância d, onde D refere-se ao diâmetro da rede. Em outras palavras, quanto mais próximos estiverem dois nós (número de saltos) menor será o erro entre seus relógios.

SOMMER e WATTENHOFER (2009) se basearam nessa ideia para propor o algoritmo GTSP (*Gradient Time Synchronization Protocol*). Trata-se de um algoritmo completamente distribuído, no qual os nós transmitem periodicamente pacotes de sincronização para seus vizinhos. Cada nó implementa um relógio lógico, baseado em seu relógio de hardware local. Inicialmente os nós ajustam as diferenças de *offset* e em seguida passam atualizar a taxa de progressão de seus relógios lógicos, de forma a atingirem um consenso local. Experimentos com 20 nós da plataforma Mica2, dispostos em anel, e simulações com número de nós e topologias variáveis demonstraram um erro médio local inferior ao encontrado nas mesmas condições com o FTSP, mantendo um erro médio global em níveis aceitáveis.

Apesar dos autores considerarem o algoritmo robusto em relação a falhas dos nós, assim como apresentado na Tabela 2.2 (RANGANATHAN e NYGARD (2010)), este trabalho provou que o GTSP é sensível a perda de mensagens de sincronismo, ou mesmo, a variação do intervalo entre essas mensagens. Os resultados desta avaliação podem ser vistos na Seção 4.3.

A seguir, apresenta-se as Tabelas 2.1 e 2.2, obtidas em RANGANATHAN e NYGARD (2010), que sumarizam o desempenho dos algoritmos descritos anterior-

mente, segundo diversas métricas.

Algoritmos	Acurácia	Eficiência Energética	Complexidade
RBS	baixa, erro médio de 29.1 us por salto	alta	alta
TPSN	16.9us por salto	alta	baixa
FTSP	1.48us por salto	alta	alta
GTSP	erro médio global e local, respectivamente 14us e 4us	alta	média
PulseSync	4.4us contra 23.96us do FTSP	alta	alta
SLTP	baixa	alta	alta

Tabela 2.1: Tabela comparativa de vários algoritmos de sincronização de relógios para RSSF

Algoritmos	Escalabilidade	Tolerância a falhas	Alinhamento de bytes
RBS	boa	não	não
TPSN	fraca	não	não
FTSP	média	não	sim
GTSP	boa	sim	não
PulseSync	boa	não	não
SLTP	alta	não	não

Tabela 2.2: Tabela comparativa de vários algoritmos de sincronização de relógios para RSSF

Capítulo 3

Dois Algoritmos Distribuídos para Escalonamento de Enlaces em Protocolos MAC

Um dos problemas mais importantes a ser solucionado por um protocolo MAC para RSSFs é reduzir o consumo de energia proveniente do subsistema de comunicação, em especial evitando ou mesmo eliminando os problemas do terminal escondido (colisões) e do *idle-listening*. Para isso, faz-se necessário o estabelecimento de um escalonamento de enlaces (*link scheduling*) ou de nós (*broadcast scheduling*) que viabilize o tráfego das mensagens com o menor custo de energia possível.

Considerando, por exemplo, o grafo da Figura 3.1, onde os vértices representam os sensores e as arestas os enlaces de comunicação, e ainda, que as transmissões são realizadas em *broadcast*, pode-se fazer a seguinte análise: se o nó A for transmitir uma mensagem para o nó B, o nó C não poderá fazer o mesmo, simultaneamente, mesmo que a mensagem não seja destinada a B. Se esta restrição não for observada, as mensagens colidirão em B. Pode-se observar, então, que se o enlace (A,B) for ativado em determinado intervalo de tempo, os enlaces (B,C), (C,D), (C,E) e (C,F) não poderão ser ativados no mesmo período. Em síntese, a condição suficiente, mas não necessária, a ser atendida para que não haja colisões é que qualquer subconjunto de nós, cuja máxima distância entre eles seja dois, não poderá ativar mais de um enlace por intervalo de tempo.

Em se tratando de um esquema TDMA com escalonamento de enlaces, o problema do *idle-listening* também é bastante reduzido, uma vez que as recepções passam a ser programadas.

O problema de se determinar um escalonamento livre de colisões que permita transmissões e recepções programadas pode ser mapeado em um problema clássico em grafos: coloração de arestas a distância-2. Neste cenário, as cores das arestas

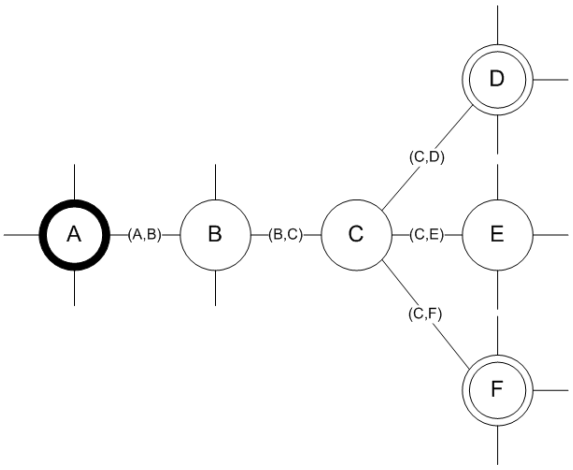

Figura 3.1: Escalonamento de enlaces

representam os intervalos de tempo de um protocolo MAC TDMA, por exemplo. Seja G um grafo não orientado, representando uma RSSF. Diz-se que duas arestas de G estão a distância-2 uma da outra se as duas são adjacentes ou se existe outra aresta que seja adjacente às duas. A coloração de arestas à distância-2 de G traduz-se na associação de cores para as arestas de tal forma que quaisquer duas arestas à distância-2 uma da outra terão cores distintas. Existem duas formas equivalentes de expressar tal requisito: *(i)* quaisquer duas arestas em um caminho consecutivo de tamanho igual a três não podem ter a mesma cor; *(ii)* no conjunto de arestas adjacentes de quaisquer nós vizinhos não pode haver um par de arestas com a mesma cor. Estas definições são suficientes para atender a restrição imposta aos enlaces de uma rede livre de colisões.

Outra abordagem do problema, seria o escalonamento de transmissões (*broadcast scheduling*), onde cada nó da rede teria um intervalo de tempo associado para realizar sua transmissão. Na verdade, trata-se de reduzir o problema anterior, de forma que todos os enlaces de um nó sejam ativados conjuntamente em uma transmissão. Sendo assim, a restrição permanece a mesma e tem perfeita ligação com o problema de coloração de vértices à distância-2.

Diz-se que dois vértices de G estão à distância-2 um do outro, se eles são adjacentes a um vértice comum (isto é, têm um vizinho em comum). Uma coloração de vértices à distância-2 de G é a associação de cores aos vértices de tal forma que quaisquer dois vértices à distância-2 um do outro tenham cores distintas. Estas cores representarão os *slots* de tempo associados aos nós.

Cabe ressaltar que, dada uma coloração de vértices à distância-2 de G, pode-se obter uma coloração de arestas à distância-2, associando-se a cada aresta o par não-ordenado das cores dos vértices incidentes à referida aresta. Com o auxílio da Figura 3.2, pode-se identificar que este mapeamento é claramente uma coloração de arestas à distância-2 do grafo, uma vez que quaisquer dois nós vizinhos não terão arestas

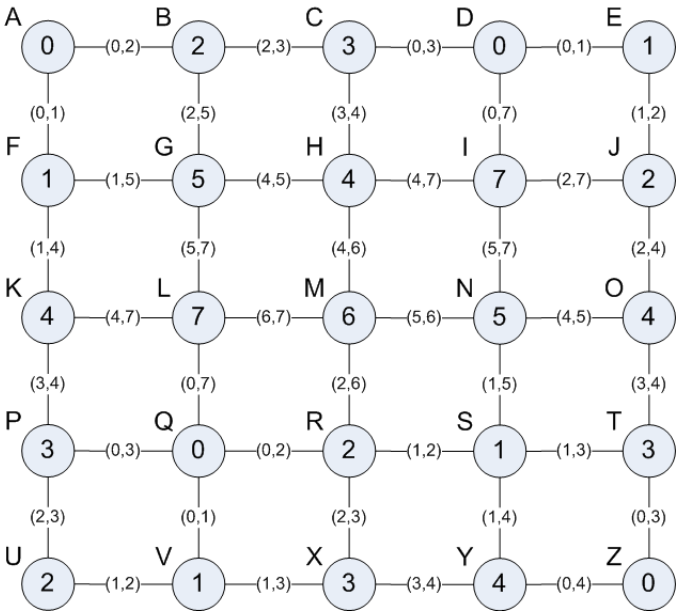

Figura 3.2: Coloração de arestas via coloração de vértices

com cores repetidas. Sendo assim, o problema da coloração de arestas à distância-2 pode ser indiretamente resolvido por meio da coloração de vértices à distância-2.

Neste capítulo propõe-se e avalia-se dois novos algoritmos probabilísticos, distribuídos e sem identificadores globais que se constituem em novas possibilidades para suportar o escalonamento de enlaces em protocolos MAC para RSSFs. Estes algoritmos são baseados na coloração de grafos (arestas e vértices) à distância-2, onde as cores representam *slots de tempo* a serem associados aos enlaces de comunicações.

Este trabalho difere dos apresentados na Seção 2.1.3 nos seguintes aspectos: (i) os algoritmos são plenamente descentralizados (não existe uma estação central ou eleição de um líder); (ii) não há necessidade de identificadores globais para os nós sensores, um indispensável requisito para RSSFs escaláveis, como mencionado em AKYILDIZ *et al.* (2002); (iii) os algoritmos são probabilísticos e totalmente independentes da topologia da rede, incluindo o caso específico de redes desconexas (isto é, o grafo de conexões é composto por múltiplas componentes conexas). É relevante notar que os algoritmos propostos neste trabalho quebram a simetria (isto é, cor das arestas) na falta de identificadores globais. Este pressuposto requer, necessariamente, uma abordagem probabilística, como demonstrado em ANGLUIN (1980).

Os dois algoritmos distribuídos propostos neste trabalho usam um mecanismo de eleição para determinar, localmente, quais nós, em cada passo da execução, têm a permissão para se colorir ou colorir suas arestas incidentes. Para o bom entendimento dos algoritmos seguem algumas definições:

- Define-se $v_i(h)$ de um nó i como o conjunto de nós que estão a distância máxima de h saltos de i;
- seja s_i um número inteiro aleatório, gerado em um nó i, a partir de uma distribuição uniforme no intervalo entre 0 e 999, utilizado para definir os nós da rede que têm permissão para colorir;
- seja $S_i(h)$ uma variável que armazena o maior valor entre s_i e s_j, sendo j o índice dos vizinhos na $v_i(h)$;
- seja r_i a lista com as cores das arestas já coloridas de um nó i;
- seja $R_i(h)$ uma lista com as cores de todas as arestas já coloridas da $v_i(h)$;
- seja e_i uma variável que denota o estado de um nó i, de acordo com as seguintes possibilidades: *ativo*, *inativo*, *vencedor* e *morto*.
- seja $E_i(h)$ uma variável booleana que controla a atividade da $v_i(h)$. Em particular, esta variável será verdadeira se houver pelo menos um vizinho na $v_i(h)$ no estado ativo.

Em resumo, os algoritmos funcionam da seguinte maneira: todo nó i envia s_i para $v_i(h)$. Todo nó k que que observar $s_k > s_j$, para todo j vizinho na $v_k(h)$ atualizará seu estado (e_k) para *vencedor*, obtendo, naquele passo de execução, o direito de colorir(suas arestas incidentes ou a si mesmo). Depois da coloração todo nó k atualiza seu estado para *inativo*, não participando mais do mecanismo de eleição, tomando parte no algoritmo apenas como um retransmissor de mensagens. Todo nó i no estado *inativo* que observar a variável $E_i(h)$ falsa, atualiza seu estado para *morto* e para.

Valores específicos serão atribuídos para o parâmetro h, de acordo com cada algoritmo proposto, o que será descrito nas próximas seções.

3.1 $Edge^3 - Sched$

O algoritmo $Edge^3 - Sched$ realiza a coloração de arestas, diretamente, à distância-2 no grafo de conexões. Relembrando, na coloração à distancia-2, em nenhum caminho consecutivo de três arestas pode haver repetição de cores.

Observando a Figura 3.1, e assumindo que as arestas *(C,D)* e *(C,F)* já estão coloridas, nota-se que quando A tenta colorir suas arestas, em particular a *(A,B)*, ele não pode fazê-lo com as cores já atribuídas as arestas *(C,D)* e *(C,F)*. Neste caso, é suficiente que o o nó A receba a informação das cores utilizadas pelo nós B e C.

Algorithm 1 $Edge^3 - Sched$

estado *A*:
nó i transmite $msg_iA = \langle r_i, s_i, e_i \rangle$
nó j recebe msg_kA, $k \in v_j(1)$, e atualiza $R_j(1)$, $S_j(1)$, $E_j(1)$
recebidas todas msg_kA, $estado \leftarrow \boldsymbol{B}$

---

estado *B*:
nó i transmite $msg_iB = \langle R_i(1), S_i(1), E_i(1) \rangle$
nó j recebe msg_kB, $k \in v_j(1)$, e atualiza $R_j(2)$, $S_j(2)$ e $E_j(2)$
recebidas todas msg_kB, $estado \leftarrow \boldsymbol{C}$

---

estado *C*:
nó i transmite $msg_iC = \langle S_i(2) \rangle$
nó j recebe msg_kC, $k \in v_j(1)$, e atualiza $S_j(3)$
if $E_i(2)$ **then**
    $estado \leftarrow \boldsymbol{D}$
else
    PARA
end if

---

estado *D*:
if $s_i > S_i(3)$ **then**
    $e_i = vencedor$
end if
    nó i transmite $msg_iD = \langle e_i \rangle$
    $estado \leftarrow \boldsymbol{E}$

---

estado *E*:
recebidas todas msg_kD, $k \in v_i(1)$
if $e_i == vencedor$ **then**
    $estado \leftarrow \boldsymbol{F}$
else if $\exists(e_k == vencedor)$ **then**
    $estado \leftarrow \boldsymbol{G}$
else
    $estado \leftarrow \boldsymbol{A}$
end if

---

estado *F*:
nó i colore suas arestas, observando $R_i(2)$
$e_i = inativo$
nó i transmite $msg_iF = \langle coloriu \rangle$
    $estado \leftarrow \boldsymbol{A}$

---

estado *G*:
nó j recebe msg_kF, k *vencedor* e $k \in v_j(1)$
    $estado \leftarrow \boldsymbol{A}$

Outra restrição que pode ser observada é que se os nós A e **E** tentarem colorir suas arestas, em particular *(A,B)* e *(C,E)*, ao mesmo tempo, existe a possibilidade que ambos os nós utilizem a mesma cor.

Diante das observações acima, infere-se que para todos os nós do grafo, o mecanismo de eleição precisa alcançar a 3-vizinhança, enquanto as informações das cores já utilizadas precisam chegar até a 2-vizinhança. A condição de parada de um nó ocorre quando a sua 2-vizinhança tem todas as suas arestas coloridas.

A implementação deste algoritmo baseou-se em uma máquina de estados que conferiu um controle assíncrono às atividades dos nós da rede. Para cada estado existe um tipo de mensagem específica. Estas mensagens serão designadas da seguinte forma: $m_i ESTADO$, onde i diz respeito ao índice do nó que irá transmiti-la e $ESTADO$ refere-se aos identificadores da máquina de estados (ver Figura 3.3). Todo nó i inicia o algoritmo com $e_i = ativo$.

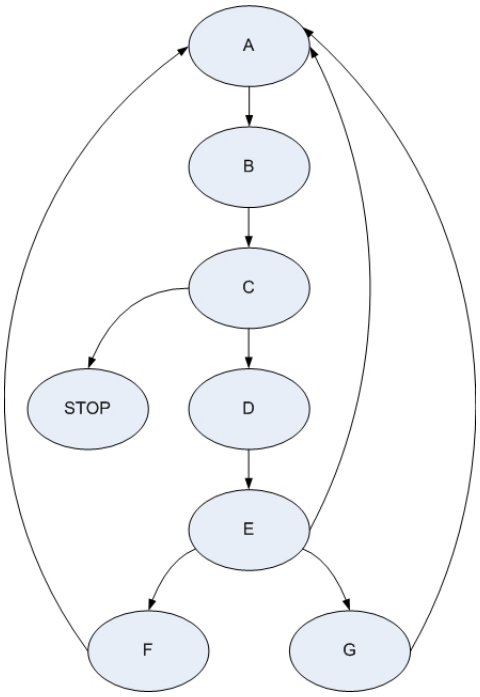

Figura 3.3: Máquina de estados que governa o funcionamento do algoritmo $Edge^3 - Sched$.

3.2 $Node^2 - Sched$

Diferente do $Edge^3 - Sched$, este algoritmo executa a coloração de arestas à distância-2 indiretamente, por meio da coloração de vértices do grafo de conexões. A Figura 3.2 ilustra como a coloração à distância-2 dos vértices de um grafo pode ser facilmente convertida em uma coloração de arestas, também à distância-2. Os valores assinalados no interior dos nós representam suas cores. Para cada aresta

atribui-se um par não ordenado das cores associadas aos nós que delimitam a respectiva aresta. Pode-se notar que o mecanismo de coloração indireta das arestas obedece a restrição imposta pela coloração direta de arestas à distância-2, qual seja: o conjunto de arestas incidentes de quaisquer par de vizinhos do grafo de comunicação não terá nenhuma cor repetida.

A máquina de estados que governa o funcionamento do algoritmo $Node^2 - Sched$ está apresentada na Figura 3.4 com sua funcionalidade descrita pelo algoritmo 2. Cabe ressaltar que a terminologia usada no algoritmo anterior permanece válida para o $Node^2 - Sched$, exceto para as seguintes variáveis:

- Seja r_i a cor do nó i;
- seja $R_i(h)$ uma lista com as cores de todos os nós já coloridos da $v_i(h)$;

Considera-se também que todo nó i inicia o algoritmo com $e_i = ativo$.

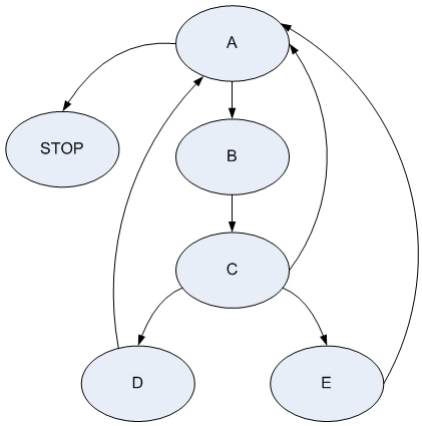

Figura 3.4: Máquina de estados do algoritmo $Node^2 - Sched$

3.3 Análise de Complexidade

Generalizando a prova apresentada em ARANTES *et al.* (2009), foi desenvolvida a seguinte análise de complexidade: seja $G = (N, E)$ o grafo de conexões para o qual será executada a coloração à distância-2, utilizando-se um dos algoritmos apresentados. Durante a execução alguns nós podem colorir suas arestas (ou a si próprios) enquanto outros aguardam a oportunidade para fazer o mesmo. Portanto define-se como nós probabilísticos aqueles que continuam executando o algoritmo e têm alguma aresta (ou eles próprios) não colorida. Complementando esta terminologia, define-se como nós determinísticos aqueles nós que já terminaram a coloração.

Define-se ainda m como a distância em saltos a ser considerada para um nó tornar-se *vencedor* e n a distância em saltos para que um nó avalie o fim da execução, tem-se a seguinte configuração para os dois algoritmos apresentados:

Algorithm 2 $Node^2 - Sched$

estado *A*:
nó i transmite $msg_iA = \langle r_i, s_i, e_i \rangle$
nó j recebe todas msg_kA, $k \in v_j(1)$, e atualiza $R_j(1)$, $S_j(1)$, $E_j(1)$
if $e_i == inativo$ **then**
    PARA
else
    $estado \leftarrow \boldsymbol{B}$
end if

---

estado *B*:
nó i transmite $msg_iB = \langle S_i(1) \rangle$
nó j recebe todas msg_kB, $k \in v_j(1)$, e atualiza $S_j(2)$
if $s_j > S_j(2)$ **then**
    $e_i = vencedor$
end if
$estado \leftarrow \boldsymbol{C}$

---

estado *C*:
nó i transmite $msg_iC = \langle e_i \rangle$
nó j recebe todas msg_kC, $k \in v_j(1)$
if $e_i == vencedor$ **then**
    $estado \leftarrow \boldsymbol{E}$
else if $\exists(e_k == vencedor)$ **then**
    $estado \leftarrow \boldsymbol{D}$
else
    $estado \leftarrow \boldsymbol{A}$
end if

---

estado *D*:
nó i transmite $msg_iD = \langle R_i(1) \rangle$ para todo nó $j|$ $(e_j == vencedor)$
nó i aguarda msg_jE, $k \in v_j(1)$
if $e_k == inativo$, para todo $k \in v_i(1)$ **then**
    $e_i = inativo$
end if
$estado \leftarrow \boldsymbol{A}$

---

estado *E*:
nó i aguarda msg_jD, $j \in v_i(1)$, e atualiza $R_i(2)$
nó i se colore, observando $R_i(2)$
if $e_k == inativo$, para todo $k \in v_i(1)$ **then**
    $e_i = inativo$
end if
nó i transmite $msg_iE == \langle e_i \rangle$ $estado \leftarrow \boldsymbol{A}$

- $Edge^3 - Sched \Rightarrow m = 3$ e $n = 2$

- $Node^2 - Sched \Rightarrow m = 2$ e $n = 1$

3.3.1 Execução

1. Todo nó i envia aos seus vizinhos $\langle r_i, s_i, e_i \rangle$;
2. todo nó i, por m vezes, processa as mensagens e as retransmite para seus vizinhos;
3. todo nó i que observar $s_i > S_i(m)$ colore suas arestas (ou a si mesmo) e deixa de ser um nó probabilístico, passando a executar somente retransmissões;
4. todo nó termina a execução quando não houver mais nós probabilísticos em $v_i(m)$.

3.3.2 Corretude

O algoritmo está correto se evitar o problema do terminal escondido e se não apresentar qualquer condição de *deadlock*. Desta forma, pode-se iniciar a análise da seguinte maneira:

1. Os nós (probabilísticos) competem para serem *vencedores* em suas h-vizinhanças usando um número de *sorteio*. Depois que um nó é declarado *vencedor*, este não compete mais (torna-se um nó determinístico). Desta maneira, probabilisticamente, todo nó se tornará um *vencedor* em algum passo da execução do algoritmo. Depois disso, todo nó aguarda que sua n-vizinhança torne-se determinística. Esta condição de parada é suficiente para assegurar que o algoritmo nunca sofrerá *deadlock* ou *starvation*.
2. Uma vez que o algoritmo assegura, para todo nó i, a associação de intervalos de tempo para transmissão das mensagens, considerando $R_i(2)$, garante-se a ausência de colisões.

3.3.3 Notação e Análise

Define-se que os algoritmos apresentados neste Capítulo completam um passo de execução, a cada passagem pelo estado inicial de suas máquinas de estados. Define-se $N_k \subseteq N$ como o conjunto de nós probabilísticos depois de k passos do algoritmo, sendo k inteiro e positivo. Tomando-se um nó $v \in N$, define-se $viz_{N_k}(v)$ como o conjunto de vizinhos probabilísticos de v, aqueles que estiverem a uma distância não superior a h saltos. Prosseguindo, define-se $G_k = (N_k, E_k)$ como um subgrafo de G induzido por k. Define-se ainda, d_i^k como o número de sorteio gerado por v_i no passo k. Os nós que tiverem o maior d_i^k em sua h vizinhança serão eleitos os *vencedores*.

1. Seja S_i^k um evento no qual v_i torna-se um *vencedor* no passo k;

2. Seja S^k a probabilidade de ocorrência de um *vencedor* no passo k;

3. Seja $B_j^k(v_i, \alpha)$ um evento que ocorre sempre que $d_i^k = \alpha > d_j^k$ para algum $v_j \in viz_{N_k}(v_i)$, o que significa que v_i vence algum nó probabilístico em sua h-vizinhança. É importante notar que $B_j^k(v_i, \alpha)$ representa eventos independentes para todo $v_j \in viz_{N_k}(v_i)$.

4. Considerando que o *sorteio* foi gerado a partir de um dado com f faces, desenvolve-se a seguinte probabilidade:

$$Pr(d_i^k = \alpha) = \frac{1}{f} \Rightarrow Pr(B_j^k(v_i, \alpha)) = \frac{\alpha}{f}; \tag{3.1}$$

5. Considere agora a probabilidade de um nó v_i ser um *vencedor*:

$$Pr(S_i^k) = \sum_{\alpha=0}^{f-1} Pr(d_i^k = \alpha) * Pr(S_i^k / d_i^k = \alpha) \tag{3.2}$$

$$Pr(S_i^k/d_i^k) = Pr(\bigcap_{v_j \in viz_{N_k}(v_i)} B_j^k(v_i, \alpha)) = \prod_{v_j \in viz_{N_k}(v_i)} (\frac{\alpha}{f}) \geq (\frac{\alpha}{f})^{\Delta_{kh}} \tag{3.3}$$

where Δ_{kh} é a quantidade de nós probabilísticos em sua h-vizinhança

6. Substituindo 3.3 em 3.2 e observando que
$Pr(d_i^k = \alpha) = \frac{1}{f}$:

$$Pr(S_i^k) \geq \sum_{\alpha=0}^{f-1} (\frac{1}{f})(\frac{\alpha}{f})^{\Delta_{kh}} \quad = \quad (\frac{1}{f^{\Delta_{kh}+1}}) \sum_{\alpha=0}^{f-1} \alpha^{\Delta_{kh}}, \forall v_i \in N_k \tag{3.4}$$

7. Usando o limite inferior apresentado em MURRAY R. SPIEGEL e LIU (2008)

$$\sum_{\alpha=0}^{f-1} \alpha^{\Delta_{kh}} \geq (\frac{(f-1)^{\Delta_{kh}+1}}{\Delta_{kh}+1} + \frac{(f-1)^{\Delta_{kh}}}{2}), \tag{3.5}$$

onde $\Delta_{kh} \geq 1$

8. Finalmente, substituindo 3.5 em 3.4 :

$$Pr(S_i^k) \geq (\frac{1}{\Delta_{kh}+1})(1 - \frac{1}{f})^{\Delta_{kh}+1} + (\frac{1}{2f})(1 - \frac{1}{f})^{\Delta_{kh}} \tag{3.6}$$

9. Considerando, como pior caso, um grafo completo com n nós:

$$Pr(S_i^0) \geq (\frac{1}{n})(1-\frac{1}{f})^n + (\frac{1}{2f})(1-\frac{1}{f})^{n-1} = (\frac{2(f-1)+n}{2fn})(1-\frac{1}{f})^{n-1} \quad (3.7)$$

$$Pr(S^k) = Pr(\bigcup S_i^k) \geq Pr(\bigcup_{i=1}^{n} S_i^0) = \sum_{i=1}^{n} Pr(S_i^0) \quad (3.8)$$

$$Pr(S^k) = (\frac{2(f-1)+n}{2f})(1-\frac{1}{f})^{n-1} \quad (3.9)$$

10. A partir daí, pode-se determinar o número médio de tentativas para que um nó seja *vencedor* no passo k:

$$T(n) = (\frac{2f}{2(f-1)+n})(\frac{f}{f-1})^{n-1} \quad (3.10)$$

11. Sendo assim, a análise de complexidade de pior caso para a convergência deste algoritmo é dada por:

$$O(f(\frac{f}{f-1})^{n-1}) \quad (3.11)$$

3.4 Método de Simulação Experimental

Para a avaliação dos dois algoritmos propostos, foi desenvolvido um ambiente de simulação. A infra-estrutura da rede de comunicação foi implementada, usando uma abordagem *multithread* na linguagem Java. Neste cenário, uma única classe é instanciada em cada nó sensor. Os métodos e atributos desta classe correspondem às funcionalidades dos algoritmos distribuídos descritos nas Seções 3.1 e 3.2. Adotou-se um modelo de comunicação *unicast* entre os nós sensores, considerando que todos os enlaces são inteiramente confiáveis.

Foram considerados dois tipos de topologias para a avaliação desses algoritmos: uma grade inteiramente regular e uma alocação aleatória dos nós sensores. Na topologia em grade, considerou-se redes compostas por 25, 49, 100, 200 e 400 nós regularmente distribuídos, de tal forma que a cobertura rádio dos nós sensores estabelecessem os enlaces apenas com os nós mais próximos (grau máximo é 4). Na alocação aleatória, os nós foram aleatoriamente distribuídos em uma área quadrada, com dimensões de 200 x 200 unidades de comprimento. Foram considerados cenários

com 150 nós, e alcance rádio variando da seguinte forma: 40, 45, 50, 55 e 60 unidades de comprimento. Assim, todos os pares de nós que se encontram a uma distância menor ou igual ao alcance rádio, um do outro, são considerados vizinhos no grafo de conexões.

Para avaliação dos algoritmos propostos, foram consideradas as seguintes métricas de desempenho: tempo de convergência médio, número médio de mensagens transmitidas, número médio de bits transmitidos e o número médio de cores utilizadas para o correto escalonamento das transmissões. O tempo de convergência foi medido em função do número de passos do algoritmo (considera-se um passo como um ciclo da máquina de estados), para que o nó termine a sua execução. Cada cenário foi simulado dez vezes. Em cada *round* da simulação acumulou-se o número de cores utilizadas, o valor médio do número de mensagens e bits transmitidos e o valor médios do número de passos. Ao final da simulação, estes valores foram divididos por dez.

3.5 Avaliação dos Algoritmos

3.5.1 Topologia em Grade

As figuras 3.5, 3.6, 3.7 e 3.8 mostram uma comparação direta do desempenho dos dois algoritmos, em termos do tempo de convergência, do número de mensagens transmitidas, número de bits transmitidos e número de cores usadas, respectivamente. Inicialmente nota-se que o algoritmo $Node^2 - Sched$ converge muito mais rápido do que o $Edge^3 - Sched$, sendo que nos cenário mais povoados, o primeiro é no mínimo 4 vezes mais rápido. Em consequência, o número de mensagens transmitidas na execução do $Node^2 - Sched$ é muito menor, chegando a um fator de 6 nos cenários mais povoados. O reflexo é semelhante na análise do número de bits transmitidos, onde os valores observado flutuam em torno de 10 vezes menor, em favor do $Node^2 - Sched$.

Por outro lado, o algoritmo $Edge^3 - Sched$ usa um número menor de cores em relação ao seu rival. Particularmente, o $Node^2 - Sched$ usa em torno de 2 vezes mais cores que o $Edge^3 - Sched$, considerando os cenários mais povoados (ver Figura 3.8. Claramente, os resultados indicam uma relação de compromisso entre o tempo de convergência e a carga de mensagens contra o número de cores utilizadas pelos algoritmos.

É importante notar também que os resultados tendem a convergir com o aumento do número de nós. Isso se deve ao fato que os algoritmos resolvem o problema da coloração localmente e assim os valores **médios** tendem a se estabilizar com o aumento da rede.

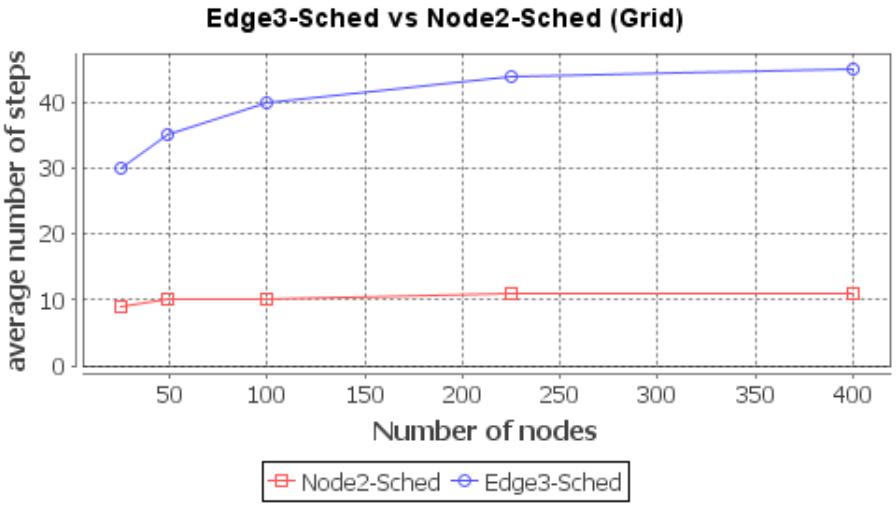

Figura 3.5: Topologia em grade: convergência

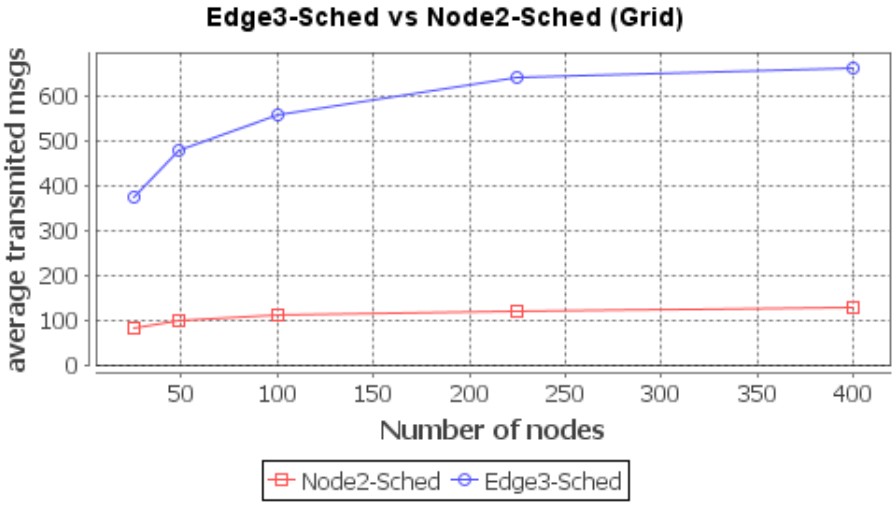

Figura 3.6: Topologia em grade: número médio de mensagens transmitidas

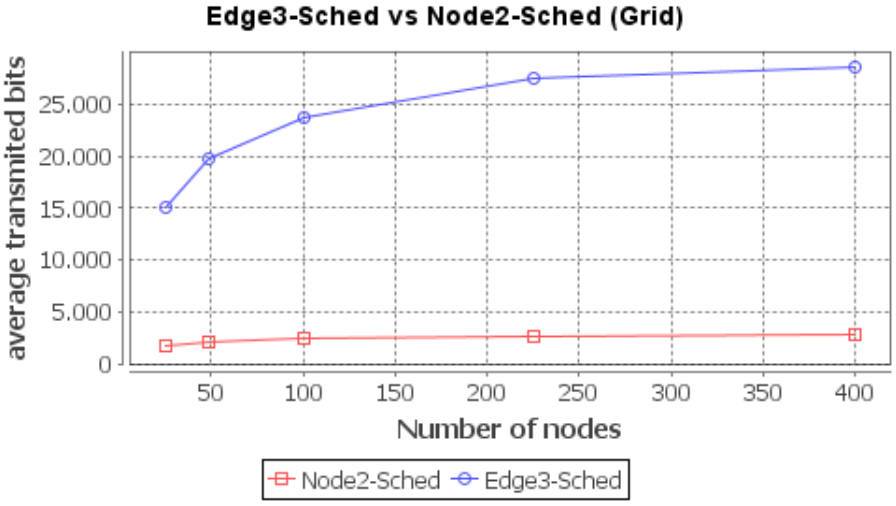

Figura 3.7: Topologia em grade: número médio de bits transmitidos

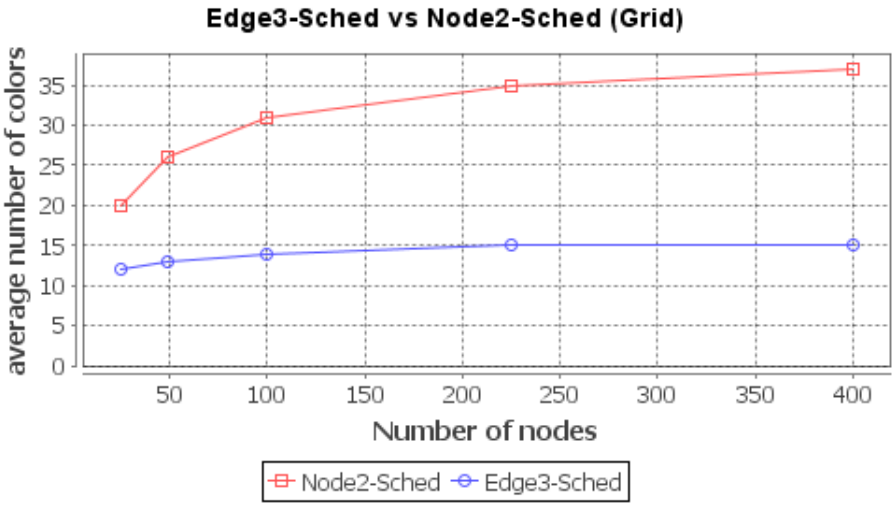

Figura 3.8: Topologia em grade: número médio de cores utilizadas

3.5.2 Alocação Aleatória de Nós Sensores

Neste cenário considerou-se redes formadas pela alocação aleatória de nós sensores, conforme descrito na Seção 3.4. Da mesma forma como foi feito na Seção 3.5.1, as Figuras 3.9, 3.10, 3.11 e 3.12 apresentam uma comparação direta do desempenho dos dois algoritmos propostos, considerando o tempo de convergência, o número de mensagens transmitidas, o número de bits transmitidos e o número de cores utilizadas, respectivamente. Os resultados foram observados em função da razão entre a área de cobertura rádio e a área de implantação da rede.

O algoritmo $Node^2 - Sched$ apresenta um melhor desempenho com relação a convergência e ao número de mensagens e bits transmitidos, ao passo que o algoritmo $Edge^3 - Sched$ usa menos cores em sua execução. Apesar desta semelhança, em relação à topologia em grade, o comportamento é diferente, no sentido que se observa uma tendencia de divergência dos resultados. As diferenças de desempenho tornam-se bem mais pronunciadas nos cenários mais densos. Curiosamente, porém, o número de cores usadas na execução do $Node^2 - Sched$ não é muito maior do que o apresentado pelo $Edge^3 - Sched$. Novamente, como observado na topologia em grade, estabeleceu-se uma relação de compromisso entre o tempo de convergência e a sobrecarga de mensagens e bits contra o número de cores utilizadas.

Como indicado no Capítulo 2, as Figuras 3.13 e 3.14 apresentam um análise comparativa entre os algoritmos até aqui apresentados e o trabalho de GANDHAM *et al.* (2008). Observou-se o número de cores necessárias para a coloração distancia-2 e o número de bits transmitidos. A comparação foi feita em cenários de topologia aleatória, considerando redes com 50, 100, 150, 300 e 400 nós, distribuídos em uma área quadrada, com dimensões de 200 x 200 unidades de comprimento e alcance rádio de 30 unidades de comprimento.

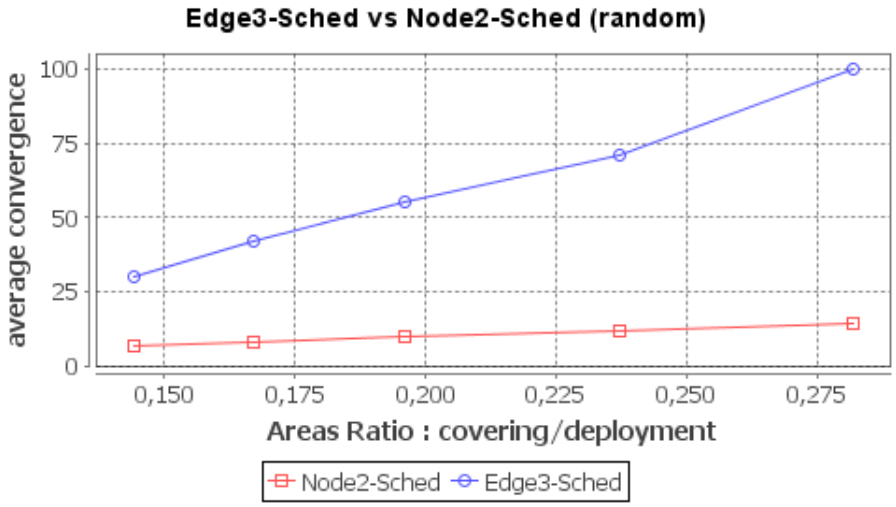

Figura 3.9: Topologia aleatória: convergência

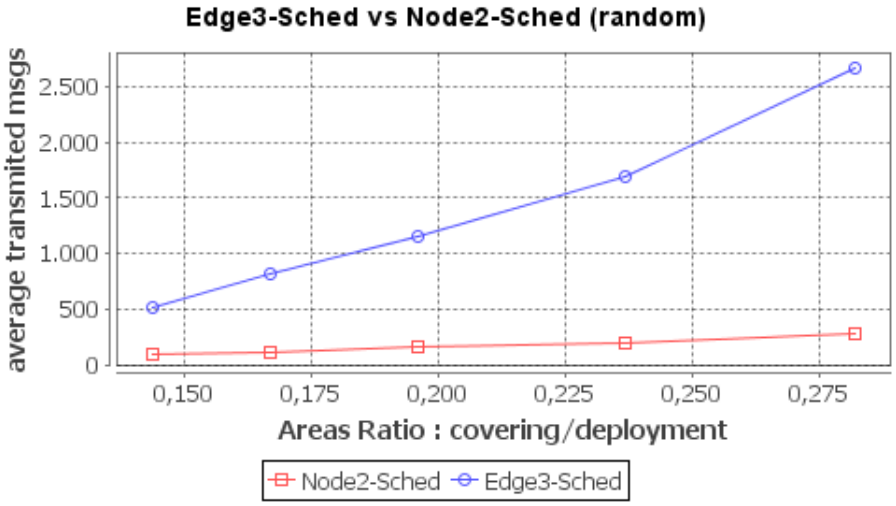

Figura 3.10: Topologia aleatória: número médio de mensagens transmitidas

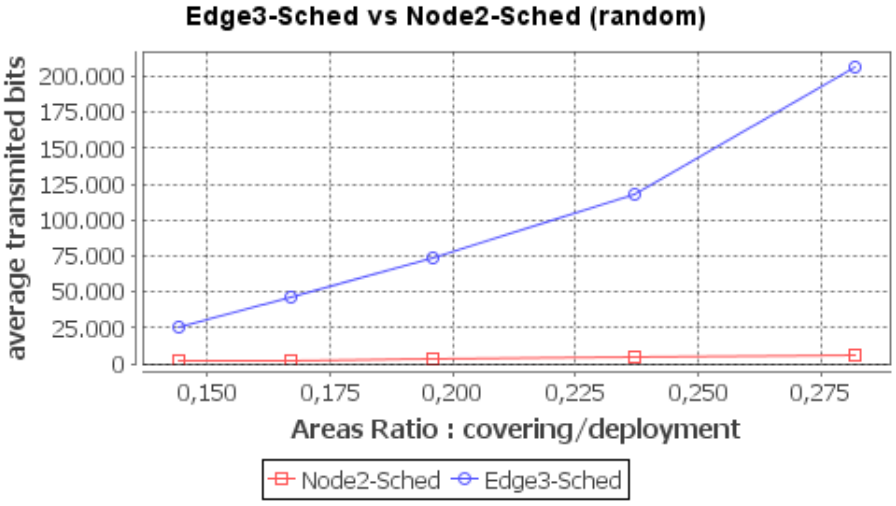

Figura 3.11: Topologia aleatória: número médio de bits transmitidos

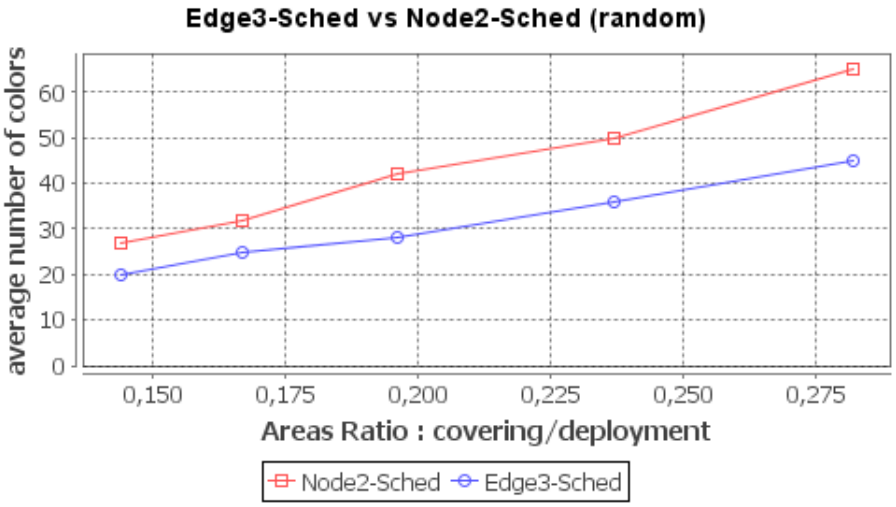

Figura 3.12: Topologia aleatória: número médio de cores utilizadas

O desempenho do algoritmo de GANDHAM *et al.* (2008), em relação ao número médio de cores utilizadas, é superior aos dos algoritmos apresentados neste Capítulo. Na verdade esta é a proposta desse algoritmo, minimizar o número de cores utilizadas. Em particular, para 400 nós, o número médio de cores utilizadas pelo $Node^2 - Sched$ é quase dez vezes maior do que o utilizado pelo algoritmo de GANDHAM *et al.* (2008). Com relação ao número de bits transmitidos, a superioridade continua em relação ao $Edge^3 - Sched$, no entanto, o algoritmo $Node^2 - Sched$ se mostra mais eficiente no número médio de bits transmitidos. Em particular, para o cenário com 400 nós, o número médio de bits transmitidos do $Node^2 - Sched$ é aproximadamente a metade do que é transmitido pelo algoritmo de GANDHAM *et al.* (2008).

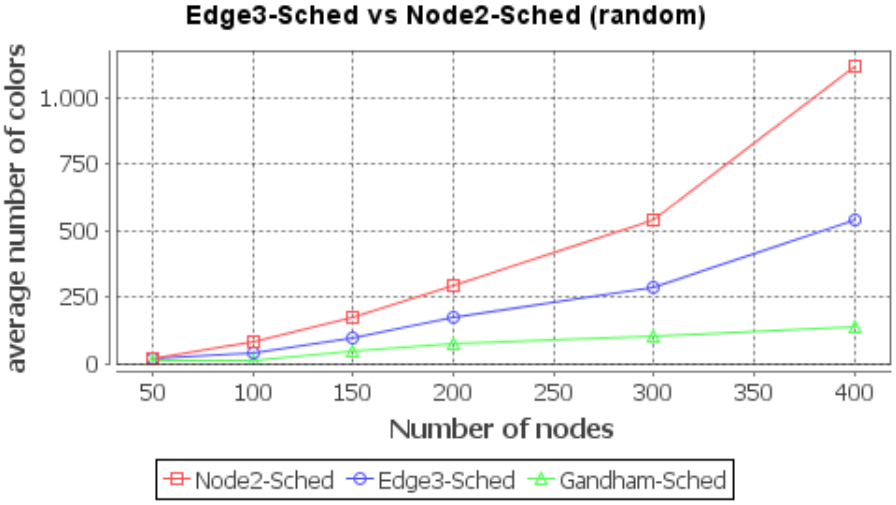

Figura 3.13: Avaliação comparativa - topologia aleatória: número médio de cores utilizadas

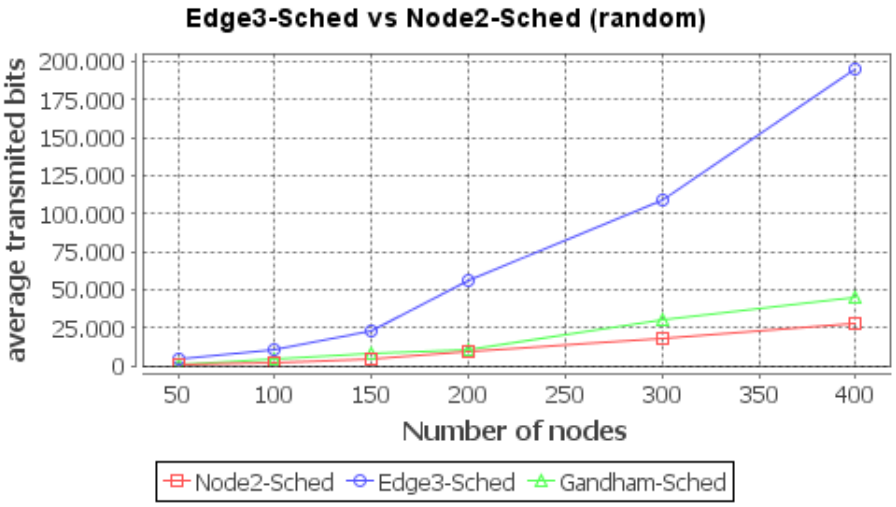

Figura 3.14: Avaliação comparativa - topologia aleatória: número médio de bits transmitidos

3.6 Discussão dos Resultados

Neste capítulo foram propostos e avaliados os desempenhos de dois algoritmos para coloração de arestas à distância-2, desenvolvidos especificamente para RSSFs. Os algoritmos são probabilísticos, distribuídos e não necessitam de identificadores globais, dando pleno suporte ao desenvolvimento de protocolos descentralizados de acesso ao meio, baseados na técnica TDMA. As simulações mostraram que o $Edge^3 - Sched$, um algoritmo de coloração direta das arestas do grafo de comunicação, tem uma maior sobrecarga no tempo de convergência e no número de mensagens e bits transmitidos, ao passo que utiliza um número menor de cores, quando comparado ao algoritmo $Node^2 - Sched$. Entretanto, os dois algoritmos servem como base para o desenvolvimento de protocolos MAC baseados no modelo de comunicação TDMA, pois dependendo dos requisitos da aplicação, não é crítico que o número de cores seja minimizado, sendo vantajoso um menor custo para a coloração de arestas.

Capítulo 4

Um novo algoritmo para sincronização de relógios em RSSF

A sincronização de relógios é um problema clássico em sistemas distribuídos que tem sido investigado por décadas em vários ambientes (LENZEN *et al.* (2009); LUNDELIUS e LYNCH (1984); SALLAI *et al.* (2006)). Este problema também é fundamental para viabilizar o bom funcionamento de inúmeras aplicações distribuídas, nas quais se faça necessária alguma forma de coordenação cronológica na interação dos nós. Em poucas palavras, duas são as razões principais que tornam este problema um desafio:

(i) apesar da taxa nominal idêntica, os relógios de hardware locais progridem a taxas reais distintas, que variam com o tempo;

(ii) as mensagens transmitidas sofrem atrasos indeterminados para chegar ao destino.

Embora na teoria nenhuma destas razões possa ser resolvida, na prática, os algoritmos distribuídos podem efetivamente lidar com eles, de maneira a prover uma razoável sincronização dos relógios.

A propriedade gradiente, introduzida em FAN e LYNCH (2004), consiste na existência de um limite superior para a diferença entre os relógios de dois nós, determinado por uma função positiva não-decrescente da distância entre estes. A distância entre os nós pode ser determinada pela incerteza no tempo de propagação das mensagens ou pelo número de saltos entre dois nós caso o tempo de propagação das mensagens seja constante ou desprezível. Um algoritmo para sincronização com gradiente de relógios deve respeitar o limite determinado por esta função para qualquer par de nós em qualquer rede e em qualquer execução.

A principal implicação desta propriedade é a formação de um gradiente da diferença entre os relógios de dois nós distintos com relação à distância entre eles. Dessa

forma a propriedade gradiente complementa a sincronização de relógios e atende aos problemas que necessitam de uma sincronização mais precisa entre nós próximos e permitem que nós mais distantes possuam uma sincronização mais dispersa.

Entre as diversas aplicações que a sincronização de relógios com propriedade gradiente é indicada, o acesso ao meio pela divisão do tempo (ex. TDMA) é certamente um deles. Neste contexto, faz-se necessária uma coordenação precisa entre nós vizinhos, de maneira a evitar os problemas relacionados na seção 1.1 da Introdução e consequentemente diminuir o desperdício de energia (por exemplo, estabelecendo-se um ciclo de trabalho para os transceptores dos nós sensores). Além disso, inúmeros mecanismos de acesso ao meio, baseados no tempo, têm sido propostos para as RSSF.

Alguns algoritmos distribuídos para sincronização de relógios com propriedade gradiente têm sido propostos, principalmente em termos teóricos. Tais propostas geralmente não consideram os detalhes necessários a uma abordagem mais realística e por isso avaliam apenas as tendências do desempenho (FAN e LYNCH (2004); PUSSENTE e BARBOSA (2009); LENZEN *et al.* (2010)). Além disso, tais propostas não consideram as características do ambiente no qual a rede entrará em operação, o que degrada o grau de confiança dos resultados.

Este trabalho concentra esforços para propor um algoritmo de sincronização de relógios com gradiente na diferença entre os relógios de uma RSSF. Algumas propostas baseadas neste conceito têm aparecido recentemente na literatura (SOMMER e WATTENHOFER (2009); SU e AKYILDIZ (2005)). A abordagem deste trabalho segue a mesma tendência (as diferenças serão discutidas detalhadamente na Seção 4.2). Propõe-se um algoritmo simples e robusto para sincronização de relógios inspirado na propriedade gradiente, *A Robust Gradient Clock Synchronization Algorithm for Wireless Sensor Networks* (RGCS), desenvolvido sobre requisitos impostos pelas RSSFs. Em particular, o RGCS é um algoritmo totalmente distribuído, que opera independentemente da topologia da rede e do mecanismo de roteamento de pacotes e não requer identificador global para os nós sensores. Além disso não há um requisito absoluto acerca da periodicidade das mensagens de sincronismo, ou seja, é permitida uma certa taxa de perda de mensagens de sincronismo. Outra característica é que os nós sensores não exigem uma referência de tempo global, apesar de tenderem a seguir de perto a evolução do tempo universal.

Em resumo, o RGCS mantém um relógio lógico local para cada nó sensor da rede. Este relógio evolui de acordo com uma taxa própria (lógica) que tem a base de tempo fornecida pelo relógio de hardware local. Os nós sensores difundem seus relógios lógicos e outros parâmetros (serão descritos detalhadamente na Seção 4.2.1). A taxa do relógio lógico (assim como o seu offset) de um nó sensor é atualizada a cada mensagem recebida. Em particular, os nós efetuam a média móvel entre taxa

de relógio lógico do vizinho e a própria. O restante da informação presente nas mensagens de sincronização permite ao algoritmo compensar os efeitos determinísticos ocorridos no próprio processo de sincronização, tal como os ajustes de offset dos relógios lógico entre duas transmissões consecutivas. Esta e outras questões serão discutidas na Seção 4.1.5. Finalmente, o RGCS é um algoritmo bastante simples e que não necessita de parâmetros a serem configurados. Para a avaliação de desempenho do algoritmo proposto, foi utilizado o *framework* Castalia (BOULIS (2010)), específico para redes de sensores e que compreende modelos de canal e de rádios muito próximos do mundo real. O Castalia é baseado na plataforma de simulação $OMNet++$ (VARGA e HORNIG (2008)). Primeiramente, o desempenho do algoritmo foi quantificado considerando-se vários cenários (topologias) e condições ideais (mensagens de sincronização periódicas e sem perdas). Os resultados foram comparados com o desempenho do GTSP, um destacado algoritmo de sincronização para RSSF, (SOMMER e WATTENHOFER (2009)). Em seguida foram impostas condições mais severas, ou seja, intervalo de mensagens variável e perda de mensagens. Nestas condições, os resultados indicaram o RGCS como uma alternativa muito mais robusta. Especificamente, para as topologias em GRID e aleatória, o erro de sincronização local praticamente não se altera para taxa de perda de mensagens de até 50%. Por fim, implementou-se um controlador para ajuste dinâmico do intervalo de mensagens em função do requisito de sincronização. O objetivo desta implementação foi assegurar o nível de sincronização desejada com a mínima frequência de mensagens de sincronismo, desonerando a rede de atividade desnecessária e consequentemente poupando energia.

4.1 Questões relacionadas à sincronização de relógios

A seguir serão apresentados os modelos de rede e de relógio para posterior discussão das questões mais importantes relacionadas à sincronização de relógios, especificamente no âmbito das RSSF.

4.1.1 Modelos: rede e relógio

Considera-se a rede de nós sensores representada por um grafo conexo e não orientado $G = (V, E)$ onde os vértices representam os nós sensores e as arestas, os canais de comunicação bidirecionais. Considera-se também um meio de propagação, tal que a transmissão de um nó $i \in V$ será recebida por todos os seus vizinhos em G. Assume-se que cada nó $i \in V$ é equipado com um relógio de hardware local $H_i(t)$. Destaca-se que $H_i(t)$ é o valor do relógio de hardware do nó i no tempo universal

t. Infelizmente, $H_i(t)$ não evolui na mesma taxa do tempo universal. Sendo assim, assume-se $h_i(t)$ como a taxa de progressão relativa do relógio de hardware, um valor geralmente variando na faixa de $[1-\epsilon, 1+\epsilon]$ para um valor bem pequeno $\epsilon > 0$. Desta forma, o valor do relógio de hardware do nó i no tempo universal t é dado por:

$$H_i(t) = \int_{t_0}^{t} h_i(\tau)d\tau + \Phi(t_0) \tag{4.1}$$

onde $\Phi(t_0)$ é o offset do relógio de hardware no tempo t_0, que pode ser interpretado como o valor do relógio de hardware no tempo t_0.

Assume-se ainda que o relógio de hardware está sujeito a imprecisão conhecida como *skew*, se $h_i(t)$ é constante em relação ao tempo. Isto significa dizer que tal relógio é consistentemente mais lento ou mais rápido do que deveria ser. O relógio de hardware também está sujeito a imperfeição denominada *drift*, se $h_i(t)$ varia com o tempo. Na prática, a faixa de variação do *drift* é bastante pequena. Sendo assim, frequentemente, a modelagem de relógios considera apenas o *skew*, ignorando os efeitos do *drift*.

Dado o modelo que acabou de ser descrito, a sincronização só é possível por meio da implementação de um relógio lógico. Sendo assim, cada nó $i \in V$ manterá um relógio lógico $L_i(t)$ progredindo com o tempo. Nota-se que $L_i(t)$ é o valor do relógio lógico mantido pelo nó i no tempo universal t. Além disso, considera-se que os nós utilizam seus relógios lógicos para disparar eventos, ou seja, um evento ocorrido no tempo universal t é disparado com $L_i(t)$ do nó i. A sincronização de relógios é executada quando os relógios lógicos de diferentes nós têm o mesmo (ou bem próximo) valor para qualquer instante do tempo universal ($L_i(t) \sim L_j(t)$, para todo $j \in V$ e para todo $t > t_0$).

Uma vez que a única referência de tempo de um nó é o seu próprio relógio de hardware, não há outra alternativa para o seu relógio lógico, que não seja progredir com base nessa referência. No entanto, diferentemente do relógio de hardware, um nó pode controlar a taxa com a qual o seu relógio lógico avança. Seja $l_i(t)$ a taxa do relógio lógico relativa à taxa de seu relógio de hardware, $h_i(t)$. Deste modo, se $l_i(t) = 1$ então a taxa do relógio lógico do nó i progride na mesma taxa que seu relógio de hardware. Na prática, para se executar a sincronização de relógios $l_i(t)$ é normalmente uma função constante por partes. Desta forma, o valor do relógio lógico do nó i é dado por:

$$L_i(t) = \int_{t_0}^{t} h_i(\tau)l_i(\tau)d\tau + \Theta(t_0) \tag{4.2}$$

onde $\Theta(t_0)$ é o offset do relógio lógico no tempo t_0, que pode ser interpretado como o valor do relógio lógico no tempo t_0.

4.1.2 Mensagens de Sincronização

Com o objetivo de sincronizar seus relógios lógicos, os nós de uma RSSF precisam trocar mensagens contendo informações sobre o relógio lógico do nó emissor. Estas mensagens, que normalmente são enviadas em *broadcast*, são usadas pelos nós que as recebem para ajuste do relógio lógico local. Muitos algoritmos assumem padrões periódicos para a transmissão destas mensagens, ou seja, cada nó transmite uma mensagem de sincronismo a cada intervalo de tempo fixo (SOMMER e WATTENHOFER (2009); SU e AKYILDIZ (2005)). Além disso, o ajuste do relógio lógico usando as mensagens recebidas é normalmente efetuado uma vez a cada período. Um exemplo são os esquemas de sincronização de relógios usando a propriedade gradiente, onde os nós buscam um consenso em relação ao tempo local. Estas abordagens, muitas vezes, impõem a necessidade dos nós esperarem a chegada das mensagens de todos os vizinhos, antes de se ajustar o relógio lógico local.

Como alertado anteriormente, em redes de sensores sem fio, a energia é considerada um dos recursos mais escassos. Sendo assim, fazer uso de um esquema específico e periódico de transmissão de mensagens de sincronização poderia ser proibitivamente custoso. Idealmente, as informações utilizadas na sincronização deveriam ser anexadas às mensagens inerentes ao tráfego da rede, sem a necessidade de uma periodicidade estrita. Ressalta-se ainda que os nós sensores estão sujeitos a falhas e que em redes de larga escala podem não haver identificadores globais únicos.

4.1.3 Os atrasos sofridos pelas mensagens

Como em qualquer modelo de transmissão de dados, existe uma latência entre o momento em que uma mensagem é criada do lado do transmissor e o instante que a mensagem é processada do lado do receptor. Este retardo é particularmente crítico para a sincronização de relógios, uma vez que, geralmente, o conteúdo das mensagens de sincronização está relacionado com o tempo. O registro dos tempos nas camadas inferiores, na MAC, por exemplo, ajuda a diminuir a incerteza, mas não elimina completamente os atrasos aleatórios e determinísticos.

Seja um nó i transmitindo o valor de seu relógio lógico para um vizinho nó j, que registra o valor do seu relógio de hardware assim que recebe a mensagem. As seguintes operações serão executadas e cada uma delas introduzirá uma parcela do atraso sofrido pela mensagem. Para esta análise, considerou-se, especificamente, as plataformas e os modos de operação praticados na maioria das redes de sensores sem fio.

1. **(EMISSOR) CPU**: requisição para transmissão;
2. **(EMISSOR) MAC**: acesso ao canal;

3. **(EMISSOR) MAC**: registro do tempo local na mensagem;
4. **(EMISSOR) RADIO**: codificação e transmissão(transmissões geram interrupções para a CPU). O retardo total depende do tamanho da mensagem do tempo para tratamento das interrupções;
5. **CANAL**: propagação do sinal de RF;
6. **(RECEPTOR) RADIO**: decodificação e tratamento de interrupção;
7. **(RECEPTOR) MAC**: leitura do relógio de hardware;
8. **(RECEPTOR) CPU**: processamento.

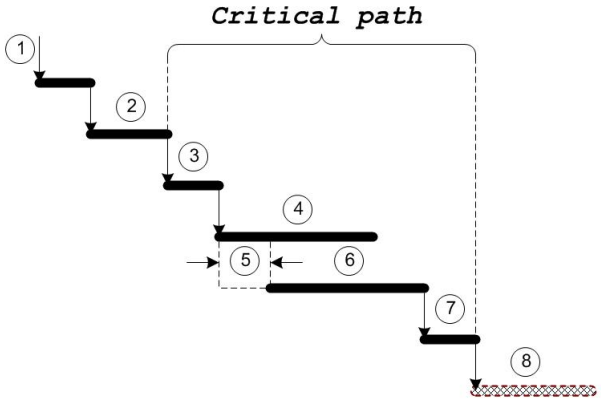

Figura 4.1: Incertezas inerentes da comunicação rádio

Na Figura 4.1 o período marcado como caminho crítico inclui as principais componentes de atrasos que devem ser considerados para minimizar a estimativa do erro entre os relógios de vizinhos. Este caminho crítico comporta os atrasos para o registro do valor do relógio lógico na mensagem do emissor (3) e para a leitura do relógio de hardware (7) no receptor. Esses atrasos são determinísticos e podem ser compensados no lado do receptor. Em RSSFs o tempo de propagação (5) de um sinal de RF é muito pequeno ($< 1us$) comparado com outros atrasos, e portanto pode ser negligenciado. Os tempos tomados para transmissão e recepção e incluem componentes determinísticas e não-determinísticas. Basicamente as componentes determinísticas correspondem ao tempo necessário para a transmissão completa de uma mensagem. Este atraso depende do tamanho da mensagem e da taxa de transmissão do rádio. A componente não-determinística do caminho crítico é função do atraso entre o instante em que o rádio gera a interrupção e o momento em que a CPU finaliza o tratamento da mesma. Este atraso é quase sempre inferior a alguns poucos microssegundos (tempo necessário para a CPU terminara a execução da instrução em curso).

Supondo-se que o relógio lógico do nó i é registrado no tempo universal t_0, a mensagem de sincronismo conterá $L_i(t_0)$. Infelizmente, conforme apresentado na Figura 4.1, a mensagem sofre uma série de retardos antes que o nó j registre o valor do seu relógio de hardware. Assumindo o atraso total sofrido pela mensagem com δ, o nó j terá o seguinte registro: $H_j(t_0+\delta)$. Nota-se que os dois valores correspondem a registros do tempo em momentos distintos. Idealmente, o nó j teria acesso a $L_i(t_0+\delta)$ ou a $H_j(t_0)$, que, em outras palavras, seria ter o conhecimento de δ. Embora na teoria este atraso seja aleatório, a cada comunicação efetuada, na prática, δ pode ser estimado com boa precisão. Analisando a composição de δ, nota-se que a sua maior componente corresponde aos tempos de transmissão e recepção da mensagem (passos 4 e 6), que tendem a ter a mesma duração, praticamente sobrepostos, a não ser pelo retardo da propagação do sinal de rádio. Além disso, uma excelente estimativa para tal latência é a razão entre o tamanho da mensagem (em *bits*) e taxa de transmissão utilizada (em *bps*). Outras origens para as componentes de atraso global, como a propagação do sinal rádio, o tratamento das rotinas de registro dos valores dos relógios, entre outros, são no mínimo uma ordem de magnitude menor nas RSSF, e por isso são frequentemente negligenciados, conforme apresentado em MARÓTI *et al.* (2004).

4.1.4 A dependência da configuração da rede

Tipicamente, o desempenho dos algoritmos de sincronização de relógios depende da estrutura topológica da rede. Além disso, alguns algoritmos têm parâmetros que precisam ser cuidadosamente escolhidos de acordo com certas características da referida topologia, tal como o diâmetro da rede (FAN e LYNCH (2004)). Outros algoritmos criam uma estrutura lógica sobre a rede para viabilizar seu funcionamento, como é o caso da formação de uma árvore de disseminação a partir da eleição de um nó líder (MARÓTI *et al.* (2004); LENZEN *et al.* (2010)). Embora essas abordagens sejam adequadas para redes cabeadas, elas são contra-indicadas para as RSSFs. Particularmente, estimar corretamente os parâmetros da configuração, assim como manter uma estrutura lógica sobre uma rede sujeita a uma série de intempéries, podem representar um custo bastante elevado, em termos de consumo de energia.

Um algoritmo de sincronização de relógios mais adequado para as redes de sensores deve operar em qualquer topologia de rede, sem a necessidade de se estabelecer qualquer estrutura lógica sobre a mesma, incluindo um cenário dinâmico onde os nós vêm e vão de acordo com o tempo. Para permitir um desdobramento fácil, o algoritmo também não deve depender da configuração prévia de qualquer parâmetro.

4.1.5 Ajustando taxa e *offset* do relógio lógico

Diversos algoritmos de sincronização de relógios ajustam o relógio lógico local ao receberem mensagens de sincronismo de seus vizinhos. Este ajuste pode modificar apenas o *offset* ou além dele, a taxa de progressão. A Figura 4.2 servirá de base para o entendimento deste conceito. A referida ilustração apresenta a evolução do relógio lógico do nó i em função do tempo universal t. O nó i recebe mensagens de sincronismo de seus vizinhos nos tempos t_2, t_3 e t_4, e executa o ajuste (*offset* e taxa) de seu relógio lógico nesses mesmos instantes. Pode-se notar que no instante t_2, apenas o offset foi ajustado. Já no instante t_3, taxa e *offset* foram atualizados. Finalmente, no instante t_4, somente a taxa do relógio lógico sofreu o ajuste. Claramente, pode-se observar que o conteúdo das mensagens recebidas determina quais os ajustes devem ser feitos. Considera-se, nesta análise, que o conteúdo das mensagens é o valor do relógio lógico de cada vizinho.

Na Figura 4.2, o nó i transmite o valor de seu relógio lógico nos instantes t_1 e t_5. Uma operação bastante comum é estimar a taxa de progressão do relógio lógico de um vizinho e usá-la para efetuar o cálculo do ajuste do relógio lógico local. Particularmente, um nó pode usar mensagens consecutivas, contendo o valor do relógio lógico de um vizinho, para estimar a taxa de progressão do relógio lógico deste vizinho, em relação ao relógio de hardware local. Considerando um nó j como vizinho do nó i, o primeiro recebe mensagens do nó i nos instantes t_1 e t_5. Ao receber estas mensagens, o nó j registra o valor de seu relógio de hardware (na verdade, um pouco atrasado em função dos retardos discutidos na Seção 4.1.3), ou seja, $H_j(t_1)$ e $H_j(t_5)$. Seja $\hat{g_{i,j}}(t)$ a estimativa da taxa de progressão do nó i em relação ao relógio de hardware do nó j, no tempo universal t. Desta forma, no instante t_5, o nó j obterá sua estimativa da seguinte forma:

$$\hat{g_{i,j}}(t_5) = \frac{L_i(t_5) - L_i(t_1)}{H_j(t_5) - H_j(t_1)} \tag{4.3}$$

Infelizmente, este procedimento pode não levar a uma estimativa precisa da taxa de progressão do relógio do nó i em relação ao relógio de hardware do nó j, no tempo t_5. Nota-se que a Equação 4.3 não leva em conta que entre as duas transmissões consecutivas do nó i ocorreram ajustes no valor de seu relógio lógico. Além disso, considerando que os ajustes de offset do relógio lógico são geralmente positivos, a Equação 4.3 tenderá a superestimar a estimativa da taxa de progressão dos relógios lógicos dos vizinhos, levando a situação em que os relógios lógicos estarão progredindo a uma velocidade mais rápida do que o tempo universal. Levando-se em conta que nas RSSF a frequência da transmissão de qualquer tipo de mensagem deve ser relativamente pequena e que o número de vizinhos tende a ser elevado, este problema pode ser maximizado de tal forma a inviabilizar o uso de tal procedimento

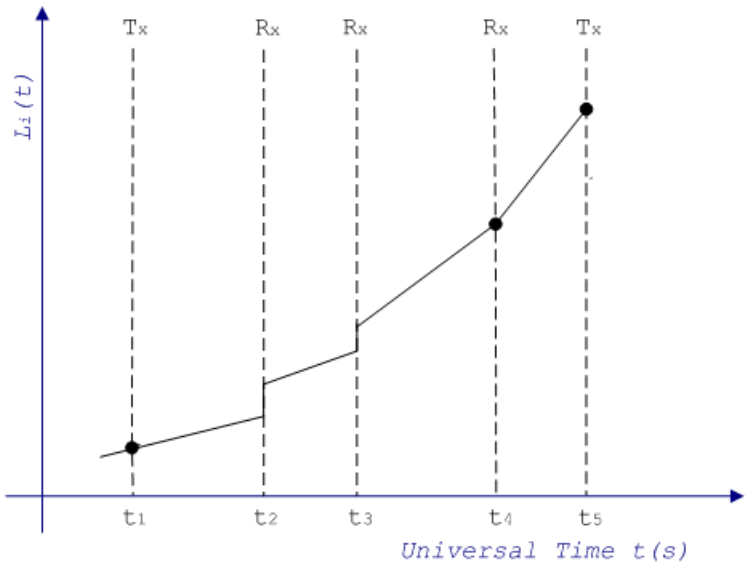

Figura 4.2: Exemplo do ajuste do relógio lógico do nó i. Mensagens de sincronismo com o valor do relógio lógico do nó i são enviadas nos tempos t_1 e t_5. O nó i recebe mensagens de sincronismo e atualiza o valor de seu relógio lógico nos tempos t_2, t_3 e t_4.

com objetivo de se estabelecer a sincronização da rede. Uma proposta para tornar mais preciso o cálculo da referida estimativa seria descontar o somatório dos ajuste de offset do numerador da Equação 4.3. Esta abordagem eliminaria a contaminação dos ajustes de offset (observar a Figura 4.3), no entanto, ainda não representaria o valor mais atual da taxa de progressão do relógio lógico, relativa ao intervalo entre t_4 e t_5. Na próxima seção, este aspecto, assim como o problema dos ajustes de offset, serão tratados no desenvolvimento de um novo algoritmo para sincronização de relógios, baseado na propriedade gradiente.

4.2 Um novo algoritmo para sincronização de relógios com propriedade gradiente

Nesta seção será apresentado um novo algoritmo de sincronização de relógios com propriedade gradiente, denominado *Robust Gradient Clock Synchronization Algorithm* (RGCS), explicando-se como cada um dos problemas levantados na Seção 4.1 foi superado. Inicia-se com uma visão geral do algoritmo, cujo funcionamento é descrito a seguir.

Cada nó envia para todos os seus vizinhos, uma mensagem de sincronismo própria ou, inclui tal informação em pacotes de outra natureza, a serem transmitidos. A referida mensagem contém o valor do relógio lógico, registrado no momento de sua transmissão, assim como outras informações que serão descritas oportunamente neste trabalho. Cada nó mantém uma variável relativa à taxa de progressão de seu relógio

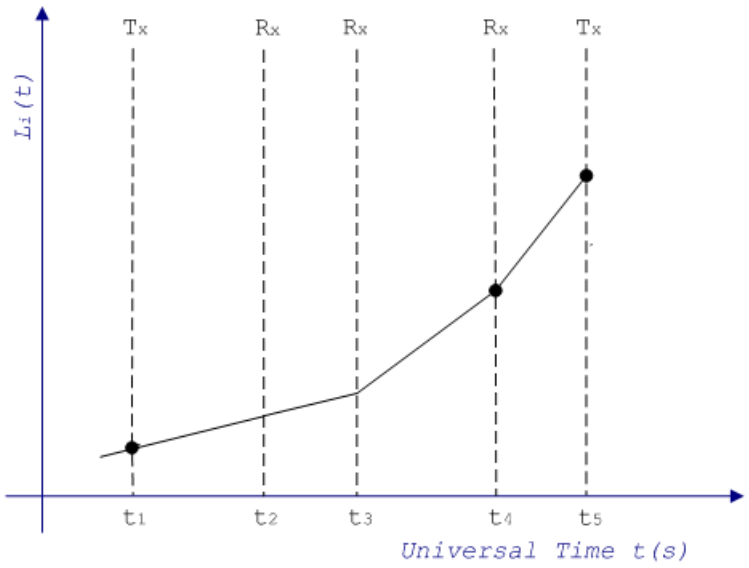

Figura 4.3: Evolução do relógio lógico do nó i, desconsiderando os ajustes de offset. Uma proposta para aumento da precisão no cálculo da estimativa da taxa de progressão do relógio lógico do nó i, em relação ao relógio de hardware do nó j

lógico, que é atualizada a cada mensagem de sincronismo recebida. Especificamente, define-se tal atualização como o resultado da média móvel entre o valor da taxa de progressão do relógio lógico local e a estimativa do valor da taxa de progressão de seu vizinho, em relação ao relógio de hardware local. Além disso, os nós podem ajustar o valor de seus relógios lógicos (*offset*) na mesma oportunidade. Em particular, se um nó recebe uma mensagem com o valor do relógio lógico de um vizinho que seja superior ao seu próprio relógio lógico, então o nó atualiza o seu relógio com o valor recebido. Pode-se notar que o algoritmo não faz qualquer suposição sobre a topologia da rede, ou a periodicidade das mensagens de sincronismo, nem requer a configuração de parâmetros específicos para guiar o seu funcionamento.

No algoritmo proposto, cada nó i mantém uma variável para o relógio lógico $L_i(t)$ que progride com base no relógio de hardware local, a uma taxa relativa $l_i(t)$ (veja a equação (4.2)). O offset e taxa do relógio lógico podem ser atualizados a cada mensagem de sincronismo recebida. Uma propriedade interessante deste algoritmo é que todos os relógios lógicos são monotônicos em relação ao tempo universal.

4.2.1 Estimando o valor atualizado do relógio lógico de um vizinho

Seja uma mensagem de sincronismo enviada pelo nó j contendo o valor de seu relógio lógico, registrado no tempo universal $t_k - \delta$, denotado por $L_j(t_k - \delta)$. Devido aos atrasos, descritos na Seção 4.1.3, o nó i registra o tempo de chegada dessa mensagem um pouco mais tarde (no caso, δ unidades de tempo), no tempo t_k. Particularmente,

o nó i obtém o valor de seu relógio lógico neste instante, que é denotado por $L_i(t_k)$. Percebe-se, claramente que os valores dos dois relógios lógicos são tomados em instantes diferentes, e por isso não podem ser comparados diretamente (veja a Figura 4.1). Por exemplo, como o nó i pode determinar se o valor de seu relógio lógico é superior ou inferior ao valor do relógio lógico do nó j no tempo t_k?

Uma alternativa seria o nó i compensar o atraso e ajustar o valor do relógio lógico do nó j, que em outras palavras significa, estimar o valor do relógio lógico de j no tempo t_k. Esta abordagem apresenta dois problemas: **(i)** o nó i não sabe com precisão o atraso sofrido pela mensagem; **(ii)** O nó i não conhece a taxa de progressão do relógio lógico do nó j em relação ao tempo universal. Nenhum desses problemas pode ser resolvido teoricamente quando os atrasos são desconhecidos (ou aleatórios), no entanto, podem ser estimados com certo grau de precisão, de forma a reduzir o impacto negativo na sincronização de relógios.

Como discutido na Seção 4.1.3, o tempo necessário para a transmissão de um pacote de dados, geralmente contribui com a maior parte do atraso sofrido por uma mensagem em redes de sensores sem fio. Sendo assim, considerando somente esta componente, pode-se reduzir consideravelmente a parte desconhecida do atraso total, levando a uma estimativa mais próxima da real. Felizmente, o atraso na transmissão é constante e pode ser calculado pelo nó receptor. Em particular, seja D_x o atraso ocasionado pela transmissão de um pacote com K bits, quando transmitido em um canal de capacidade igual a B bps. Equacionando estes parâmetros, tem-se que $D_x = K/B$, uma operação facilmente executável pelo nó receptor, desde que o mesmo tenha conhecimento do tamanho da mensagem, K, e da capacidade do canal, B.

Para que o nó i possa estimar a taxa com a qual o relógio lógico do nó j evolui, a mensagem de sincronismo deverá incluir a referida taxa tomada no momento em que a mensagem for construída, ou seja, $l_j(t_k - \delta)$. É importante notar que esta taxa é tomada em relação ao próprio relógio de hardware do nó j e não em relação ao tempo universal (veja a equação (4.2)). No entanto pode-se admitir, sem perda de confiança, que a taxa de progressão do relógio de hardware é muito próxima da taxa real (tempo universal), especialmente quando comparada com a taxa do relógio lógico, que pode ser variavelmente arbitrária. Sendo assim, assume-se que o relógio lógico de um nó j progride com a taxa $l_j(t_k - \delta)$ em relação ao tempo universal.

Usando a taxa de progressão do relógio lógico do nó j, pode-se, agora, compensar o atraso de transmissão e atualizar o valor de seu relógio lógico, imediatamente após o seu recebimento. Em particular, quando o nó i recebe uma mensagem de sincronismo enviada pelo nó j no tempo t_k, contendo $L_j(t_k - \delta)$ and $l_j(t_k - \delta)$, o nó

i efetua a seguinte atualização:

$$\hat{L_j}(t_k) = L_j(t_k - \delta) + l_j(t_k - \delta)D_x \tag{4.4}$$

É importante notar que $\hat{L_j}(t_k)$ é uma estimativa do valor do relógio lógico do nó j no tempo t_k a qual pode ser muito mais precisa que $L_j(t_k - \delta)$. Sempre que necessário, este será o procedimento para computar o valor atualizado do relógio lógico de um nó vizinho.

4.2.2 Alcançando os vizinhos mais adiantados

No algoritmo proposto, quando um nó recebe uma mensagem de sincronismo e efetua o ajuste do valor do relógio lógico de seu vizinho de acordo com a equação 4.4, é realizada uma comparação direta entre os valores absolutos dos relógios lógicos de ambos os nós. Caso o valor do relógio lógico local seja inferior ao do vizinho, imediatamente o relógio lógico local sofre um ajuste de *offset* de forma a alcançar o valor do relógio lógico do vizinho. Intuitivamente, esta abordagem permite que relógios mais lentos ou muito atrasados cheguem a um consenso mais rápido com sua vizinhança. Sendo assim, quando o nó i recebe uma mensagem de sincronização do nó j no tempo t_k, imediatamente registra o valor de seu relógio lógico $L_i(t_k)$. Em seguida, o nó i calcula a estimativa do valor do relógio lógico no tempo t_k, de acordo com a Equação 4.4 e compara este valor com o valor registrado de seu próprio relógio lógico. Se $L_i(t_k) < \hat{L_j}(t_k)$, então o nó i ajusta o valor do seu relógio lógico para $\hat{L_j}(t_k)$.

4.2.3 Ajustando as taxas dos relógios lógicos com média móvel

Relembrando que cada nó i atualiza sua taxa de progressão de relógio lógico, $l_i(t)$, a cada mensagem de sincronismo recebida. Primeiramente o nó i efetua o cálculo da estimativa da taxa de progressão do relógio lógico do nó j em relação ao relógio de hardware local, ou seja, $\hat{l_j}(t_k)$ (o cálculo dessa estimativa será descrito na Seção 4.2.4). Usando essa estimativa, o nó i atualiza a taxa de progressão de seu relógio lógico da seguinte maneira:

$$l_i(t_k) = l_i(t_{k-1})(1 - \rho(t_k)) + \hat{l_j}(t_k)\rho(t_k) \tag{4.5}$$

onde $l_i(t_{k-1})$ é a taxa de progressão do relógio lógico do nó i quando da última atualização efetuada no tempo t_{k-1}, e $0 < \rho(t_k) < 1$ é o parâmetro que reflete a importância relativa entre a taxa anterior do relógio lógico do nó i e a estimativa de

taxa de progressão do relógio lógico do nó j. A equação (4.5) é uma média móvel exponencialmente balanceada, atualizada a cada mensagem de sincronismo recebida com o parâmetro $\rho(t_k)$.

A escolha de $\rho(t_k)$ tem um papel importante na velocidade de convergência de consenso entre os relógios lógicos. No entanto, escolher um valor constante para $\rho(t_k)$ não garante um desempenho ótimo do algoritmo. Sendo assim, foi desenvolvida uma estratégia para calcular dinamicamente o referido parâmetro, considerando quatro cenários distinguidos pelo valor e taxa de progressão dos relógios lógicos, conforme a descrição a seguir.

1. O relógio lógico do nó i está atrasado e mais lento em relação ao seu vizinho. Isto significa que $L_i(t_k) < \hat{L}_j(t_k)$ e que $l_i(t_{k-1}) < \hat{l}_j(t_k)$. Neste caso o nó i deve priorizar a taxa de progressão do nó j. Assim, $\rho(t_k)$ é calculado da seguinte maneira:
$$\rho(t_k) = \frac{\hat{l}_j(t_k)}{l_i(t_{k-1}) + \hat{l}_j(t_k)} > 0.5 \tag{4.6}$$

2. O relógio lógico do nó i está adiantado e mais rápido em relação ao seu vizinho. Isto significa que $L_i(t_k) > \hat{L}_j(t_k)$ e que $l_i(t_{k-1}) > \hat{l}_j(t_k)$. Neste caso o nó i deve priorizar a taxa de progressão do nó j. Assim, $\rho(t_k)$ é calculado da seguinte maneira:
$$\rho(t_k) = \frac{l_i(t_{k-1})}{l_i(t_{k-1}) + \hat{l}_j(t_k)} > 0.5 \tag{4.7}$$

3. O relógio lógico do nó i está atrasado e mais rápido em relação ao seu vizinho. Isto significa que $L_i(t_k) < \hat{L}_j(t_k)$ e que $l_i(t_{k-1}) > \hat{l}_j(t_k)$. Neste caso o nó i deve priorizar a taxa de progressão do nó j. Isto pode parecer não intuitivo, mas cabe ressaltar que o nó i também avançará o seu relógio lógico para o valor do relógio lógico do nó j, uma vez que está atrasado em relação ao mesmo. Além disso, como o relógio lógico de i é mais rápido, deve-se priorizar a taxa de progressão do relógio lógico do vizinho j para prevenir um distanciamento excessivo de i. Assim, $\rho(t_k)$ é calculado da seguinte maneira:
$$\rho(t_k) = \frac{l_i(t_{k-1})}{l_i(t_{k-1}) + \hat{l}_j(t_k)} > 0.5 \tag{4.8}$$

4. O relógio lógico do nó i está adiantado e mais lento em relação ao seu vizinho. Isto significa que $L_i(t_k) > \hat{L}_j(t_k)$ e que $l_i(t_{k-1}) < \hat{l}_j(t_k)$. Neste caso o nó i deve priorizar a sua própria taxa de progressão. Assim, $\rho(t_k)$ é calculado da seguinte maneira:
$$\rho(t_k) = \frac{l_i(t_{k-1})}{l_i(t_{k-1}) + \hat{l}_j(t_k)} < 0.5 \tag{4.9}$$

4.2.4 Estimando a taxa de progressão do relógio lógico de um vizinho

Conforme destacado na seção anterior, o algoritmo RGCS requer o cálculo da estimativa da taxa de progressão de um nó vizinho em relação ao relógio de hardware local. Este procedimento, discutido na Seção 4.1, é considerado um desafio porque o relógio lógico pode ser contaminado por ajustes de *offset* e taxa no intervalo entre duas transmissões consecutivas da mensagem de sincronismo. A seguir será apresentado, um modelo de solução para esta questão.

Para facilitar o entendimento, toma-se as seguintes definições: seja t_k e t_n os instantes da $k-esima$ mensagem transmitida, e $n-esima$ mensagem recebida, ambas em relação ao nó i. Assume-se que os relógios de hardware estão sujeitos ao erro de *skew*, ou seja, variação em relação a taxa nominal de progressão, mas não apresentam *drift*, e as taxas de progressão dos relógios lógicos têm comportamento linear. A ideia para se tratar os efeitos causados pelas mudanças sofridas pelo relógio lógico do nó i no intervalo entre duas transmissões consecutivas serão apresentadas em seguida, com o auxílio da Figura 4.4.

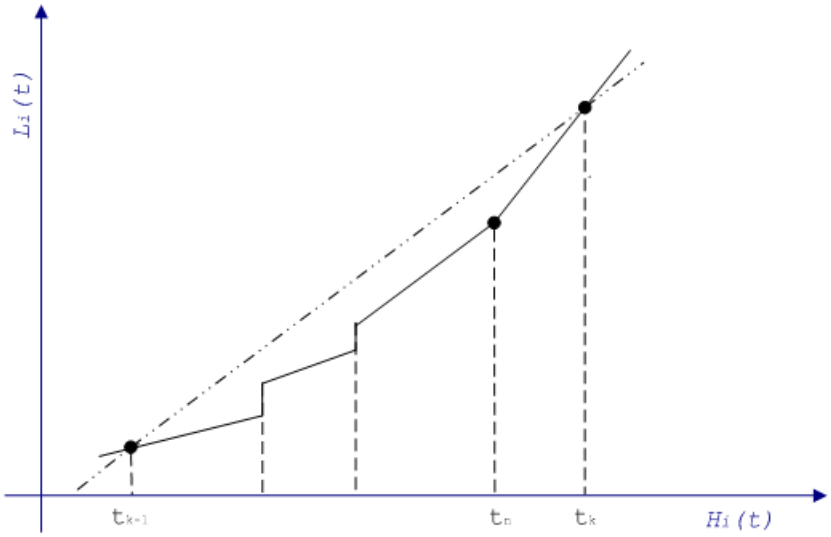

Figura 4.4: Exemplo do ajuste do relógio lógico do nó i. Mensagens de sincronismo com o valor do relógio lógico do nó i são enviadas nos tempos t_1 e t_5. O nó i recebe mensagens de sincronismo e atualiza o valor de seu relógio lógico nos tempos t_2, t_3 e t_4.

O nó i conhece o valor da taxa de progressão de seu relógio lógico no instante t_k e também pode calcular a variação média desta taxa no período entre t_{k-1} e t_k. O nó i incorpora estes dois valores na mensagem de sincronismo que será enviada ao nó j. Ao receber a referida mensagem, o nó j calculará a estimativa da taxa de progressão do relógio lógico do nó i, em relação ao seu relógio de hardware, considerando o período $[t_{k-1}, t_k]$. Em seguida, o nó j usará a razão entre as duas

taxas incorporadas à mensagem para compensar as possíveis mudanças ocorridas no relógio lógico do nó i, no mesmo período. A seguir, os detalhes deste procedimento: O nó i pode calcular $g_{i,i}(t_k)$ (a variação média da taxa de progressão de seu relógio lógico) para o intervalo $[t_{k-1}, t_k]$, que é o mesmo intervalo a ser usado pelo seu vizinho j. Assim, tem-se:

$$g_{i,i}(t_k) = \frac{L_i(t_k) - L_i(t_{k-1})}{H_i(t_k) - H_i(t_{k-1})} = \frac{L_i(t_k) - L_i(t_{k-1})}{h_i\,(t_k - t_{k-1})} \tag{4.10}$$

O nó i também conhece a taxa de progressão de seu relógio lógico no instante t_k, denotada por $l_i(t_k)$, e podendo ser calculada da seguinte maneira:

$$l_i(t_k) = \frac{L_i(t_k) - L_i(t_n)}{H_i(t_k) - H_i(t_n)} = \frac{L_i(t_k) - L_i(t_n)}{h_i\,(t_k - t_n)} \tag{4.11}$$

Define-se, então, $s_i(t_k)$ como a razão entre estas duas taxas de progressão do relógio lógico do nó i no instante t_k. E, desta forma:

$$s_i(t_k) = \frac{l_i(t_k)}{g_{i,i}(t_k)} = \frac{L_i(t_k) - L_i(t_n)}{L_i(t_k) - L_i(t_{k-1})}.\frac{t_k - t_{k-1}}{t_k - t_n} \tag{4.12}$$

É importante notar que $s_i(t_k)$ é calculada usando apenas as informações disponíveis para o nó i. Sendo assim, bastaria a transmissão desta relação no lugar de $g_{i,i}(t_k)$ e $l_i(t_k)$. No entanto, cabe lembrar que o RGCS faz ajuste de taxa e *offset*, sendo este último realizado depois da compensação do seu atraso na transmissão. Para este cálculo faz-se necessário o conhecimento da taxa de progressão do relógio lógico do vizinho, no instante t_k, o que corresponderia a $l_i(t_k)$, no caso do nó i sendo o transmissor.

Assim, ao receber a mensagem de sincronismo de i, o nó j calula a estimativa da taxa de progressão do relógio lógico do nó i em relação ao seu relógio de hardware, e usando $s_i(t_k)$ faz a conversão de domínios($H_i \rightarrow H_j$) da taxa atualizada do nó i, ou seja, no instante t_k. Esta operação elimina a necessidade do nó j conhecer as mudanças ocorridas no relógio lógico do nó i. Na verdade, esta operação compensa qualquer variação ocorrida no período compreendido entre duas transmissões consecutivas. Tal procedimento, no entanto, não garante a precisão plena, em virtude do retardo sofrido pela mensagem de sincronismo. Em outras palavras, o nó j registra os valores de seu relógio de hardware com pequenos atrasos, denotados como $t_{k-1} + \delta_1$ e $t_k + \delta_2$, onde δ_i, $i = 1, 2$, é o retardo total sofrido pela primeira e segunda mensagens, medidos na escala de tempo universal. De qualquer maneira, o nó j pode estimar a taxa de progressão do relógio lógico do nó i da seguinte forma:

$$\hat{g}_{i,j}(t_k) = \frac{L_i(t_k) - L_i(t_{k-1})}{H_j(t_k + \delta_2) - H_j(t_{k-1} + \delta_1)}.s_i(t_k) \tag{4.13}$$

É importante salientar que neste cálculo, o nó j usa o valor de $L_i(t)$ que foi incluído na mensagem de sincronização pelo nó i, ao invés de fazer uso do valor compensado pelo atraso na transmissão (ver Equação 4.4. Além disso, tal estimativa é simplesmente multiplicada por $s_i(t_k)$, doravante denominado fator de correção. Continuando o equacionamento, tem-se o seguinte:

$$\hat{g}_{i,j}(t_k) = \frac{L_i(t_k) - L_i(t_n)}{h_j\,(t_k - t_n)} . \frac{t_k - t_{k-1}}{t_k - t_{k-1} + \delta_2 - \delta_1} \tag{4.14}$$

Assim, $\hat{g_{i,j}}(t_k)$ é uma estimativa bastante precisa da taxa de progressão mais atual do relógio lógico do nó i (no tempo t_k) em relação ao relógio de hardware do nó j (pode-se notar que este valor é equivalente ao $\hat{l}_i(t_k)$ apresentado na subseção anterior). Particularmente, se $\delta_2 = \delta_1$ ou se tais valores são conhecidos, a estimativa passa a ser exata. Na prática, δ_1 e δ_2 não são conhecidos, mas geralmente são pequenos e bastante semelhantes. Sendo assim, ignorando completamente δ_1 e δ_2 na Equação (4.13) chega-se a uma estimativa $\hat{g_{i,j}}(t_k)$, bem mais precisa que a originalmente calculada, por meio da Equação 4.3. Considera-se, ainda, que mesmo se os relógios de hardware sofressem o efeito do erro de *drift*, tal estimador ainda seria o mais adequado, uma vez que os erros gerados por tal perturbação causariam variações muito pequenas, em relação aos ajustes de offset, ocorridos entre duas transmissões consecutivas.

Finalmente, como alertado anteriormente, o método descrito acima também compensa os ajustes de offset, isto ocorre, porque na conversão dos domínios ($H_i \rightarrow H_j$), somente o período entre t_n e t_k é considerado e os ajustes de offset, já estão computados em t_n. Sendo assim, a Equação (4.14) resolve adequadamente as questões ilustradas pela Figura 4.2.

Resumo do algoritmo RGCS

O algoritmo 3 resume a operação de difusão das mensagens de sincronismo, que deve ser feita periodicamente por todo nó da rede. No entanto, este período não precisa ser determinístico ou realmente confiável, e não precisa ser idêntico para todos os nós, conforme discussão mais adiante. O algoritmo 4 resume a computação efetuada a cada mensagem de sincronismo recebida.

Algorithm 3 Nó i transmite uma mensagem de sincronismo para todos os seus vizinhos

seja t o tempo universal atual
atualizar $L_i(t)$
calcular $s_i(t)$
difundir a mensagem de sincronismo $\langle L_i(t), l_i(t), s_i(t) \rangle$

Algorithm 4 A cada momento que o nó i recebe uma mensagem de sincronismo do nó j

seja t o tempo universal atual
atualizar $L_i(t)$
calcular $\hat{L}_j(t)$
if $\hat{L}_j(t) > L_i(t)$ **then**
  $L_i(t) \leftarrow \hat{L}_j(t)$
end if
calcular $\rho(t)$
calcular $\hat{g}_{i,j}(t)$
atualizar $l_i(t)$

4.3 Avaliação

Nesta seção serão apresentados os resultados da avaliação de desempenho do algoritmo proposto, RGCS, usando o framework Castalia (BOULIS (2010)). Este simulador é baseado na plataforma OMNeT++ e foi especificamente desenvolvido para simulação de algoritmos e/ou protocolos considerando-se modelos realísticos dos rádios e das condições de propagação. Foram consideradas as seguintes condições para realização dos experimentos:

- cada nó da rede é modelado de forma independente dos outros;
- assume-se que os relógios de hardware podem sofrer um desvio na taxa de progressão nominal (skew), que no caso foi ajustada para 921 KHz. Em função disso, cada relógio de hardware teve sua taxa real (e fixa, ou seja os relógios não sofrem o efeito denominado drift, que é a variação da taxa com o tempo) escolhida de uma faixa uniformemente distribuída em torno da taxa nominal, com erro máximo de 30 ppm;
- cada nó da rede entra em operação com o relógio de hardware zerado, em um intervalo inicial, variando entre 0 e 30 segundos no tempo universal.

O modelo do rádio escolhido foi o CC2420 da Texas Instruments, compatível com protocolo ZigBee no padrão IEEE 802.15.4 - 2.4 GHz, cujos parâmetros de configuração estão listados a seguir:

- taxa de transmissão: 250 Kbps
- modulação: PSK
- banda do canal: 20 MHz
- sensibilidade: -95dBm

- potência de transmissão: 0 dBm
- consumo de potência nos estados RX e TX: 62 mW
- consumo de potência no estado SLEEP: 1.4 mW

As mensagens têm um tamanho de 72 bytes (incluindo os dados de sincronização e os cabeçalhos das camadas física e de enlace. As colisões são tratadas, simplesmente, descartando os pacotes. Juntamente com o atraso fixo da propagação, que é compensado pelo algoritmo proposto, todo pacote sofre um atraso aleatório introduzido na recepção do rádio para representar os atrasos provenientes do tratamento de interrupções. Este atraso foi modelado como uma distribuição exponencial com média 2 microssegundos. Finalmente, considerou-se que dois nós são vizinhos, se eles podem receber mensagens um do outro (ou seja, existe um enlace sem fio entre eles). Com relação aos parâmetros dos rádios citados acima, pode-se afirmar que dois nós serão vizinhos se estiverem afastados um do outro a no máximo 10m.

Os cenários escolhidos dizem respeito a quatro topologias: linha, anel, grade e aleatória. Na topologia aleatória, os nós são colocados uniformemente distribuídos dentro de uma área quadrada de lado Q, que é escolhido de forma a manter a densidade da rede constante conforme o aumento do número de nós. Particularmente, a densidade utilizada foi de 0.022 nós/m^2, implicando, na média, que cada nó terá 6 vizinhos. Os dois principais parâmetros desta avaliação são o número de nós n e o intervalo de transmissão de mensagens de sincronismo P.

A principal métrica considerada foi o erro médio local, que é definido como a média das diferenças médias dos relógios lógicos de cada nó e os da sua vizinhança no tempo universal t. Cada diferença média entre um nó e sua vizinhança é obtida calculando-se o erro médio quadrático. Assim, tem-se:

$$e_l(t) = \frac{1}{|E|} \sum_{(i,j)\in E} |L_i(t) - L_j(t)| \tag{4.15}$$

onde E é o número de enlaces da rede. Nota-se que o erro médio local muda com o tempo t. No entanto, este erro geralmente converge para um valor suficientemente grande de t. Denota-se tal valor como a média dos erros médios quadráticos, com se segue. Seja T um valor grande o suficiente para t medido em segundos, e seja S um número elevado de amostras. Assim, define-se:

$$\overline{e}_l = \frac{1}{S} \sum_{i=1}^{S} e_l(T + 25 * i) \tag{4.16}$$

Nos experimentos apresentados nesta seção, T foi geralmente escolhido em torno de 750K para as topologia em linha e anel, e em torno de 200K para as topologias

em grade e aleatória. Esta variação se fez necessária em função da convergência mais lenta do erro médio local nas topologias em linha e anel. Para os resultados apresentados foi usado $S = 1000$.

Também foi reportado o erro médio global, que é definido como a média dos erros médios quadráticos entre cada nó e todos os outros da rede. Assim:

$$e_g(t) = \frac{1}{\binom{|V|}{2}} \sum_{i,j \in V} |L_i(t) - L_j(t)| \tag{4.17}$$

onde V é o conjunto de nós da rede. Como feito anteriormente, define-se o tempo médio do erro médio global como $\overline{e}_g$. Nota-se que o erro médio global não considera a topologia da rede, enquanto o $e_l(t)$ computa apenas os erros entre vizinhos. Assim, $\overline{e}_l$ é mais adequado para avaliar a propriedade do gradiente de um algoritmo de sincronização de relógios.

Finalmente, cada valor reportado nas avaliações que se seguem representa a média de 10 rodadas independentes da simulação.

4.3.1 Verificação do gradiente de erro entre relógios

Com o objetivo de investigar o comportamento do erro entre relógios em função da distancia d (em número de saltos) entre os mesmos e do diâmetro da rede foram realizados quatro experimentos com redes na topologia em linha, cada uma com 50, 100, 150 e 250 nós. Em cada um destes experimentos foi monitorado o erro médio de pares de nós distantes entre si 5, 10, 15, 20 e 25 saltos. Os resultados podem ser observados nas Figuras 4.5 a 4.6.

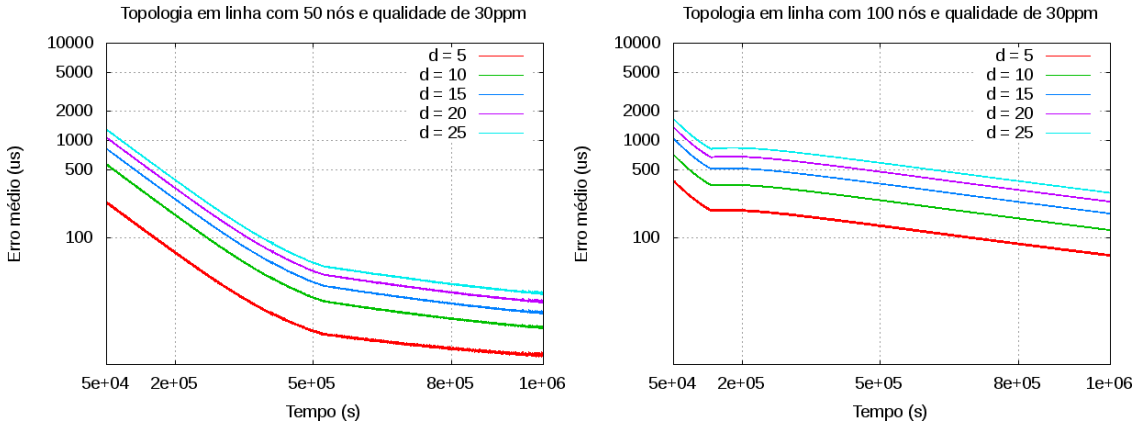

Figura 4.5: Erro médio em função da distancia d entre os nós. Diâmetros de 50 e 100 saltos.

Os resultados apresentados nas Figuras 4.5 a 4.6 confirmam a intuição de que o erro entre os relógios da rede se comporta na forma de um gradiente, em função

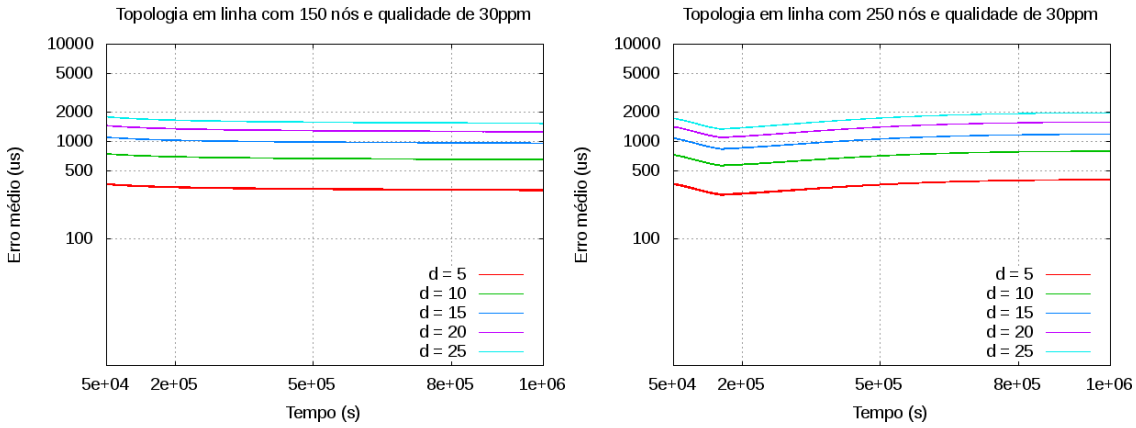

Figura 4.6: Erro médio em função da distancia d entre os nós. Diâmetros de 150 e 250 saltos.

da distância entre os nós. Particularmente, nos resultados da Figura 4.5 pode-se observar a forte influência do diâmetro da rede no erro entre os relógios. Com o aumento do diâmetro (150 e 250 saltos), os erros tendem a convergir para valores bem próximos, com uma influência menor do diâmetro da rede, conforme apresentado na Figura 4.6.

4.3.2 Comunicação segura e intervalo entre mensagens constante

Nesta seção seção serão apresentados os resultados do desempenho do RGCS em comparação ao GTSP (SOMMER e WATTENHOFER (2009)). Para tal, o algoritmo GTSP foi implementado no mesmo *framework* Castalia, de acordo com a descrição do algoritmo (assim como informações obtidas por meio de contato com seus autores). Foram observadas pequenas diferenças entre os resultados das duas implementações para as mesmas condições de simulação (topologia e parâmetros). Esta diferença, no entanto, é perfeitamente compreensível, uma vez que o simulador usado neste trabalho utiliza modelos para as camadas física e de enlace bem mais próximos da realidade, refletindo um desempenho um pouco abaixo do apresentado em SOMMER e WATTENHOFER (2009).

A Figura 4.7 apresenta o comportamento do erro médio local em função do tempo para várias topologias ($n = 100, P = 90s$). Pode-se observar que o tempo de convergência e o valor médio estabilizado estão relacionados com o diâmetro da rede e o tamanho da vizinhança. Nota-se que as topologias em grade e aleatória (diâmetro menor e com mais vizinhos) convergem bem mais rapidamente e para um valor menor, tanto no RGCS quanto no GTSP. No entanto o desempenho do RGCS é melhor no tempo de convergência e com um valor médio de $e_l(t)$ muito abaixo do apresentado pelo GTSP. O mesmo comportamento é observado para a topologia em

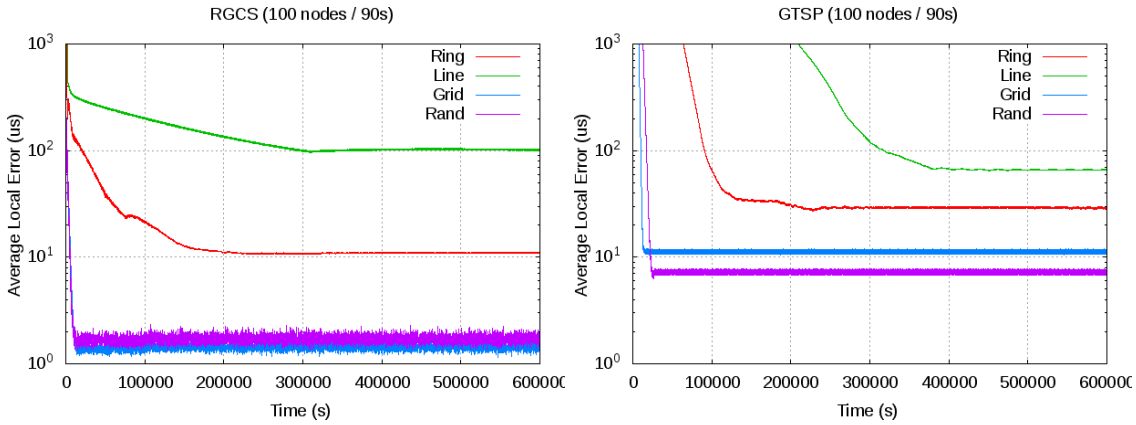

Figura 4.7: Erro médio local em função do tempo, medido nos algoritmos RGCS e GTSP.

anel.

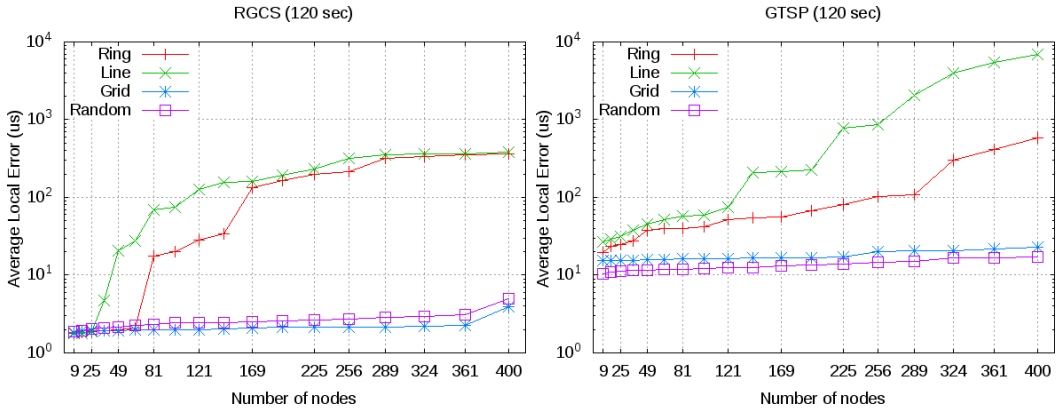

Figura 4.8: Erro médio local em função do número de nós, medido nos algoritmos RGCS e GTSP.

A Figura 4.8 o erro médio local $\overline{e}_l$ para várias topologias de rede, em função do número de nós da rede, com $P = 120$s (cabe ressaltar que o eixo y está em escala logarítmica). Observa-se que para as topologias em grade e aleatória, tanto o RGCS quanto o GTSP apresentam medidas para o erro médio local ligadas diretamente ao número de nós da rede, apesar do RGCS chegar a um valor médio cinco vezes menor, no mínimo, considerando todos os tamanhos da rede. Para as topologias em linha e anel, O RGCS mostra uma taxa de crescimento no $\overline{e}_l$ até 256 nós, ponto a partir do qual o valor médio parece convergir, independente do aumento de n. O mesmo não acontece com o algoritmo GTSP, no qual o $\overline{e}_l$ continua crescendo, indicando que o erro possa não convergir com o aumento de n.

A Figura 4.9 mostra o comportamento do erro médio local ($\overline{e}_l$) em função do intervalo de mensagens de sincronização (P) para várias topologias com 100 nós cada. Novamente, observa-se que para as topologias em grade e aleatória, a taxa

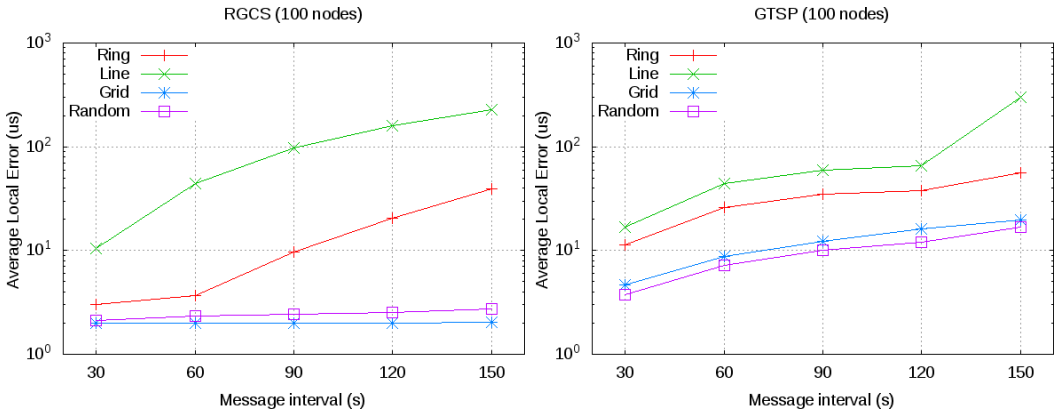

Figura 4.9: Erro médio local em função do intervalo entre mensagens de sincronização, medido nos algoritmos RGCS e GTSP.

de crescimento do erro médio local no algoritmo RGCS cresce suavemente com P, variando entre 30s e 150s. A avaliação do algoritmo GTSP, nas mesmas condições, revela que o valor médio alcançado é da ordem de cinco vezes maior do que o apresentado pelo RGCS. Para as topologias em linha e anel, o aumento do erro é significante para ambos os algoritmos, indicando que tais topologias são bastante sensíveis a intervalos de mensagens grandes.

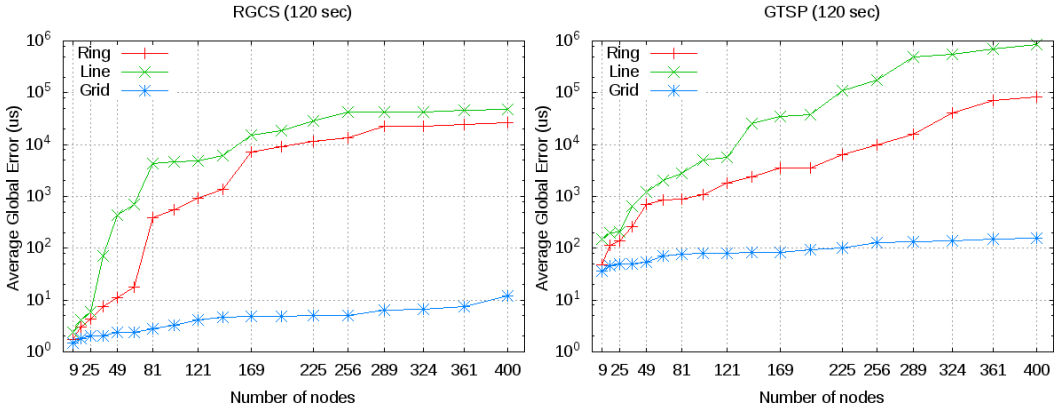

Figura 4.10: Erro médio global em função do número de nós, medido nos algoritmos RGCS e GTSP.

A Figura 4.10 apresenta o comportamento do erro médio global $\overline{e}_g$, medido em função do número de nós n. Observa-se uma tendência similar ao que foi apresentado em relação ao erro médio local para ambos os algoritmos RGCS e GTSP, conforme a Figura 4.8. No entanto, cabe destacar a diferença em ordem de magnitude do eixo das ordenadas na avaliação dos dois algoritmos, revelando um desempenho bastante superior a favor do RGCS. É interessante salientar que o erro médio global apresentado pelo algoritmo RGCS na topologia em grade é o menor que o erro médio local do algoritmo GTSP na mesma topologia, (comparar com a Figura 4.8). Tendência semelhante foi observada nos outros casos, indicando que o erro médio

local do algoritmo RCGS não aumenta ao custo do aumento do seu erro médio global.

4.3.3 Comunicação sujeita a falhas e intervalo entre mensagens aleatório

A partir de agora, a robustez do RGCS será avaliada considerando dois diferentes cenários de sujeição a falhas. No primeiro cenário, os nós escalonam a próxima transmissão com um tempo escolhido a partir de uma distribuição normal com média 60s e desvio padrão 6s. Nota-se que embora o intervalo de mensagens seja aleatório, todos os nós têm o mesmo intervalo médio, ou seja, P é independente e identicamente distribuído. Além disso, a transmissão das mensagens pode falhar com uma probabilidade p. Em outras palavras, a mensagem de sincronização pode ser descartada antes da transmissão com uma probabilidade p. Foram consideradas redes com 100 nós.

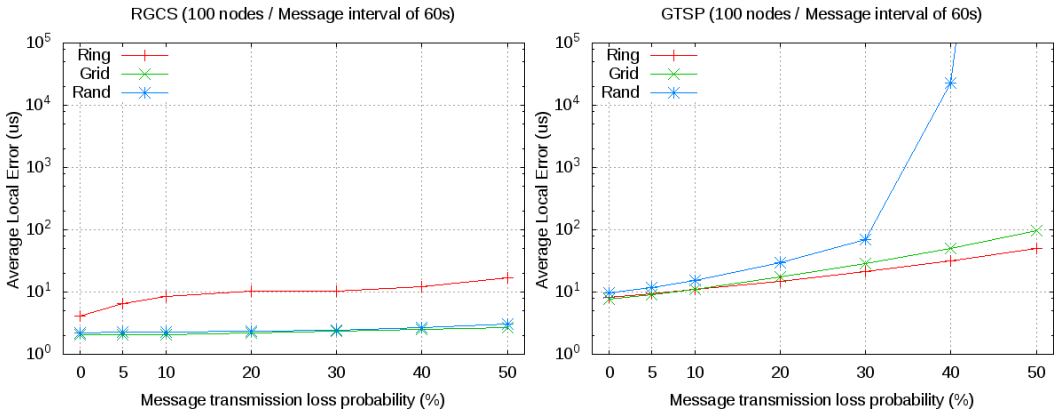

Figura 4.11: Desempenho do RGCS e do GTSP em função da probabilidade de falha na transmissão.

A Figura 4.11 apresenta o comportamento do erro médio local para diferentes topologias e várias probabilidades de falha na transmissão. Foi observado que o RGCS é bastante robusto em relação às falhas. Particularmente, para as topologias em grade e aleatória, o algoritmo apresenta uma degradação bastante suave para probabilidades de falha variando entre 0% e 50%. Já o algoritmo GTSP, apresenta um comportamento bem diferente, no qual a degradação de desempenho se mostra bastante elevada e atrelada ao aumento de p. Além disso, nota-se que em topologias onde os nós têm uma vizinhança maior, os resultados pioraram, em oposição ao que foi observado para o RGCS. Em especial, para a topologia aleatória, o algoritmo GTSP simplesmente diverge para uma probabilidade de falha na transmissão de 50%. O fraco desempenho do GTSP ocorre em função da necessidade do algoritmo esperar o recebimento das mensagens de todos os seus vizinhos antes de atualizar

seu relógio lógico local. Esta situação ainda é agravada pelo fato do cálculo da estimativa da taxa de progressão dos vizinhos ser realizada com valores de relógios lógicos que podem estar significativamente defasados em relação ao tempo universal.

Na abordagem anterior, todos os nós tinham o mesmo intervalo médio para transmissão de mensagens de sincronismo. Em um segundo cenário para avaliação da robustez do algoritmo, considera-se o relaxamento desta suposição, de maneira a atribuir diferentes intervalos entre mensagens de sincronismos para os nós da rede. Em particular, o intervalo médio entre mensagens é escolhido uniformemente de um intervalo aleatório $[30, B]$ no momento em que o nó entra em operação na rede. Seja P_i o intervalo entre mensagens de sincronismo do nó i. As mensagens serão enviadas periodicamente de acordo com uma distribuição normal com média P_i e desvio padrão de $0.1P_i$. Considerou-se um valor crescente de B para avaliar o desempenho dos algoritmos em função de quão diversos são os valores médios dos intervalos de mensagens de sincronismo.

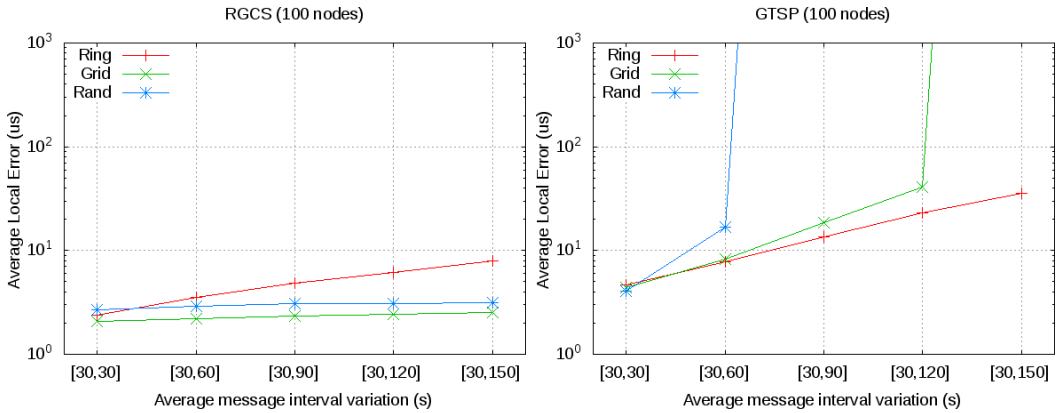

Figura 4.12: Desempenho do RGCS e do GTSP em função da faixa de variação do intervalo médio entre mensagens de sincronismo.

A Figura 4.12 apresenta o comportamento de $\overline{e}_l$ para diferentes topologias, considerando diferentes faixas de variação do intervalo médio entre mensagens de sincronismo. Nota-se que o algoritmo RGCS também é bastante robusto quando avaliado sob estas condições. Especificamente, para as topologias em grade e aleatória, o desempenho do algoritmo quase não é influenciado para faixas variando entre $[30, 30]$ to $[30, 150]$, indicando que o RGCS é praticamente insensível a irregularidade dos intervalos de transmissão dos nós da rede. Novamente, o desempenho do algoritmo GTSP é notadamente diferente, revelando o aumento do $\overline{e}_l$ com o parâmetro B. Mais uma vez, nota-se que para as topologias em grade e aleatória, o GTSP simplesmente não converge para valores grandes de B, indicando que o algoritmo não é robusto em relação à diversidade nos intervalos de mensagens dos nós da rede.

4.4 O ajuste dinâmico do intervalo de mensagens de sincronismo

As seções anteriores mostraram o bom desempenho do algoritmo RGCS, quando submetido a diversas condições relativas à transmissão das mensagens de sincronização. Na seção 4.3.2 observou-se o comportamento do algoritmo diante de um modelo de comunicação totalmente confiável e um intervalo constante entre mensagens de sincronização. Já na seção 4.3.3, avaliou-se o desempenho do algoritmo com um modelo de comunicação sujeito a falhas e intervalos entre mensagens de sincronismo distintos para cada nó da rede. Constatou-se que o RGCS tem um desempenho muito bom, mesmo quando submetido a rigorosas condições de funcionamento. No entanto, em alguns cenários de RSSF a frequência de sensoriamento é muito baixa, podendo reproduzir situações nas quais a atividade da rede se resume a manutenção da sincronização. Esta observação traz à tona a seguinte questão: qual o máximo intervalo entre mensagens de sincronismo necessário para garantir a tolerância desejada para o erro entre os relógios lógicos da rede? Procurando uma resposta para tal questão, chegou-se a a conclusão de que a situação ideal seria que o algoritmo ajustasse dinamicamente o intervalo de mensagens de sincronização de forma a atingir um requisito previamente estabelecido, que seria a máxima tolerância de erro entre os relógios de nós vizinhos. Esta abordagem permite a minimização do custo da sincronização para a rede. O grande desafio que se apresenta é modelar o mecanismo de controle, que ajusta o intervalo entre as mensagens de sincronismo.

4.4.1 O estudo do mecanismo de controle

A modelagem de mecanismos de controle é uma disciplina muito conhecida na Engenharia Elétrica com extensa aplicação na automação de sistemas. O modelo de controle consagrado na literatura é o denominado controlador PID, cujo significado da abreviatura é o seguinte: **P**roporcional, **I**ntegrador e **D**erivativo. A Figura 4.13 será útil para o entendimento do modo de operação deste mecanismo de controle.

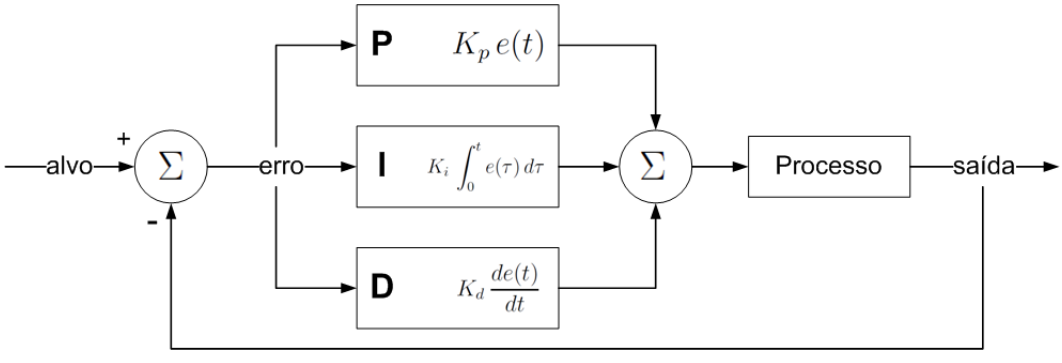

Figura 4.13: Diagrama básico de um controlador PID

Um controlador PID é basicamente uma malha de realimentação, na qual se monitora o erro (diferença entre a medida de uma variável do processo e o valor desejado para a mesma), com o objetivo de minimizá-lo, ajustando as entradas de controle de processo. A função de controle deste mecanismo é composta por três mecanismos de controle distintos: proporcional , integrador e derivativo. Heuristicamente , esses mecanismos podem ser interpretados em termos do tempo: P depende do valor atual do erro, o I depende do acúmulo dos erros do passado e D representa a predição de erros futuros com base na atual medida de variação. A soma ponderada dessas três ações é usada para ajustar o processo. Algumas aplicações podem exigir o uso de apenas uma ou duas ações para prover o controle apropriado do sistema. Isto é feito, atribuindo-se o valor 0 aos pesos das componentes de controle a serem desprezadas. Um controlador PID com essa característica poderá ser chamado de PI, PD, P. Estes controladores podem ser contínuos ou discretos. No primeiro caso, a atuação do controle é contínua, ou seja, a resposta a uma variação do erro depende apenas do retardo do controlador. No controlador discreto, a variável do sistema é periodicamente comparada com o valor desejado para obtenção do erro. Neste caso o intervalo entre as amostragens da variável do sistema influi diretamente no tempo de resposta do controlador. Acompanhando a Figura 4.14, tem-se que $e(kT_a)$ é a k-ésima amostra do erro, que é medido em intervalos de tempo T_a. A função $u(kT_a)$ é a saída do controlador e representa a k-ésima atuação do controlador nos sistema.

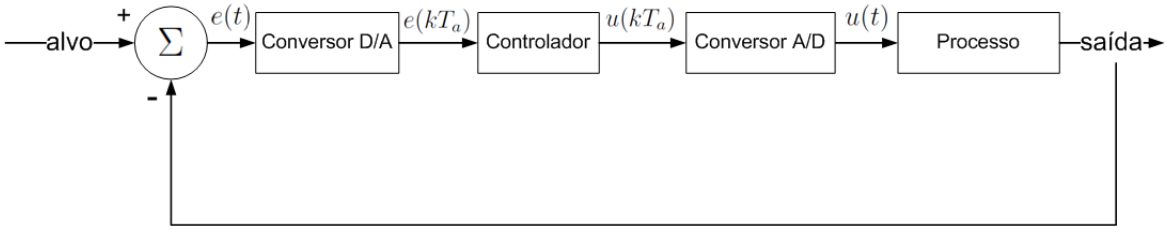

Figura 4.14: Diagrama de um controlador PID discreto

Trazendo os conceitos descritos anteriormente para o cenário pretendido, ou seja, maximizar o intervalo de mensagens de sincronismo de forma a manter o requisito desejado de sincronização dos relógios lógicos, devem ser feitas as seguintes considerações:

1. A atuação do controlador PID, apresentada na literatura é baseada no modelo centralizado. Já nas RSSF, é frequentemente desejável que o controle seja distribuído nos nós da rede;

2. O controlador discreto é o mais adequado para a implementação no cenário supracitado, uma vez que as oportunidades de se efetuar o controle correspondem aos eventos de recepção de mensagens de sincronismo;

3. Com relação a referida proposta de controle, monitora-se o erro de sincronização para aumentar ou diminuir o período entre mensagens de sincronismo. É importante notar que a atuação do controlador influencia na sua própria frequência de atuação. Exemplificando: se o erro diminuir, o período entre mensagens de sincronismo aumentará e, com isso, o intervalo para a próxima atuação do controle também será maior. Considerando tal situação, se houver uma inversão, e o erro começar a subir, alcançando um valor proibitivo, o controlador poderá não apresentar a resposta em tempo oportuno.

4. A ideia de se diminuir o período entre mensagens de sincronismo para forçar a diminuição do erro de sincronização, ou vice-versa, é correta ao se pensar na rede como um todo. No entanto, não se pode fazer a mesma consideração pensando-se na atuação de um nó individualmente. Quando um nó percebe que a diferença entre seu relógio lógico e os de seus vizinhos está aumentando, provavelmente é ele que ainda não alcançou um consenso com a vizinhança. Aumentar a frequência das mensagens forçaria a toda a vizinhança concordar com a taxa dele próprio. Esta não parece ser uma boa prática, uma vez que na abordagem consensual prevalece a tendência da maioria. Sendo assim, o que se pretende, ao identificar o aumento do erro, é na verdade aumentar a atividade local e da vizinhança, para que haja uma convergência mais rápida dos relógios lógicos.

4.4.2 A modelagem de alto nível

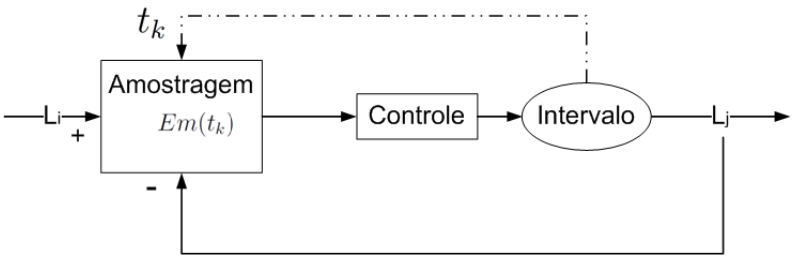

Figura 4.15: Modelo de controlador PID adotado para controlar o intervalo entre mensagens de sincronismo.

A malha de controle desenvolvida para controlar o intervalo entre as mensagens de sincronismo teve que ser adaptada em relação ao modelo apresentado na Figura 4.15, em função das considerações feitas anteriormente. Na verdade, as três componentes de controle (P, IeD) serão utilizadas para regular o ajuste dinâmico, no entanto, não serão combinadas de forma ponderada, mas selecionadas segundo outro procedimento. Antes de se explicar tal funcionamento é necessário que sejam feitas

algumas definições, as quais estão apresentadas a seguir. A Figura 4.16 auxiliará o entendimento.

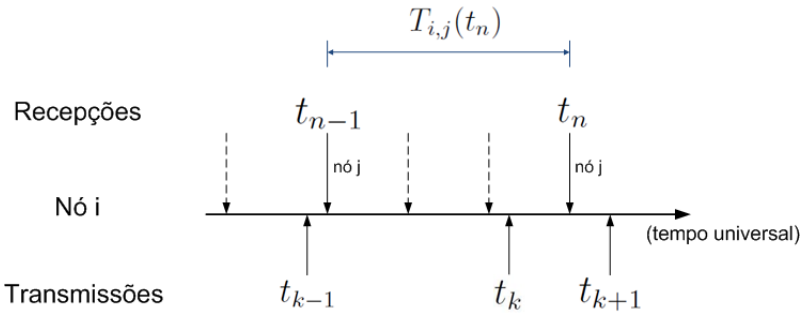

Figura 4.16: Cronologia dos eventos de transmissão e recepção de mensagens de sincronismo.

- Seja δ o parâmetro a ser escolhido como tolerância do erro entre relógios lógicos de nós vizinhos;
- seja $T_i(t_k)$ o intervalo entre a k-ésima e a (k-1)-ésima transmissões de mensagem de sincronismo efetuadas pelo nó i, medido no relógio de hardware do nó i;
- seja $T_{i,j}(t_n)$ o intervalo entre a n-ésima e a (n-1)-ésima mensagens de sincronismo recebidas de um vizinho j, medido (intervalo) no relógio de hardware local (nó i);
- seja $\overline{T}_{i,j}(t_n)$ o valor médio de $T_{i,j}(t_n)$;
- seja $e_i(t_n)$ o valor absoluto da diferença entre o relógio lógico local (nó i) e o valor do relógio lógico de um vizinho, obtido na n-ésima recepção de mensagens de sincronismo;
- seja $Em(t_k)$ o valor médio dos $e_i(t_n)$, registrados no período entre a k-ésima e a (k-1)-ésima transmissões de mensagem de sincronismo;
- seja $P(t_k)$ uma função linear que reporta um valor entre $[-0.5, 0.5]$ em função de $Em(t_k)$;
- seja $I(t_k)$ uma função integradora modelada como a média móvel entre $I(t_{k-1})$ e $Em(t_k)$;
- seja $D(t_k)$ uma função derivadora modelada como a média móvel entre $D(t_{k-1})$ e $\frac{Em(t_k)-Em(t_{k-1})}{t_k-t_{k-1}}$;

Implicitamente, o algoritmo RGCS realiza a medição do intervalo entre mensagens de sincronização enviadas pelos seus vizinhos. Esta operação é feita a cada mensagem recebida, por hora do cálculo da estimativa da taxa de progressão dos relógios lógicos de seus vizinhos. Esta medição é na verdade o denominador da Equação (4.3), ou seja, todo nó pode monitorar a variação do intervalo entre mensagens de sincronismo de seus vizinhos. Desta maneira mantém-se a variável $\overline{T}_j(t_n)$ para computar a média móvel dos intervalos medidos. Assim:

$$\overline{T}_j(t_n) = \overline{T}_j(t_{n-1}) * \beta + T_j(t_n) * (1 - \beta) \tag{4.18}$$

onde $T_j(t_n) = H_i(t_n) - H_i(t_{n-1})$ e β é o coeficiente da média móvel. Nas implementações realizadas neste trabalho, considerou-se $\beta = 0.5$.

O erro médio observado entre duas transmissões consecutivas é calculado da seguinte forma:

$$Em(t_k) = \frac{\sum_{a=1}^{z} (|L_i(t_n) - L_j(t_n)|)}{z} \tag{4.19}$$

onde $L_i(t_n)$ denota o valor do relógio lógico local e $L_j(t_n)$ o valor do relógio lógico de um vizinho, ambos registrados no momento da recepção da n-ésima mensagem de sincronismo pelo nó i. A variável z representa o número de mensagens recebidas. É importante salientar que para este cálculo, somente serão computadas as mensagens recebidas entre a k-ésima e (k-ésima -1) transmissões realizadas pelo nó i.

A função integradora $I_e(t_k)$ é uma média móvel em relação aos valores de $Em(t_k)$, sendo obtida conforme a Equação 4.20

$$I(t_k) = I(t_{k-1}) * \beta + Em(t_k) * (1 - \beta) \tag{4.20}$$

Definiu-se um intervalo de confiança para o retorno desta função da seguinte forma: sob certas condições (vide o algoritmo (6), valores retornados dentro do intervalo $[0.9 * \delta, 1.1 * \delta]$ caracterizam que a sincronização está em um patamar adequado, conforme a tolerância desejada.

A função derivadora $D_e(t_k)$ também é obtida por meio de uma média móvel, em relação a variação temporal (tempo medido no relógio de hardware local) de $Em(t_k)$, observada a cada transmissão de mensagem de sincronismo. A Equação (4.21) sintetiza este raciocínio:

$$D(t_k) = D(t_{k-1}) * \beta + \frac{Em(t_k) - Em(t_{k-1})}{H_i(t_k) - H_i(t_{k-1})} * (1 - \beta) \tag{4.21}$$

Finalmente a função linear $P_e(t_k)$ é calculada com base em $Em(t_k)$, da seguinte

maneira:

$$P(t_k) = \frac{-Em(t_k) * 0.5}{\delta} + 0.5 \tag{4.22}$$

O comportamento desta função pode ser observado na Figura 4.17. Para que esta tenha um comportamento simétrico, restringiu-se o máximo valor de $Em(t_k)$ a duas vezes o valor de δ. Pode-se notar que se $Em(t_k)$ for superior a δ, a função retorna um valor negativo, caso contrário, o retorno é positivo.

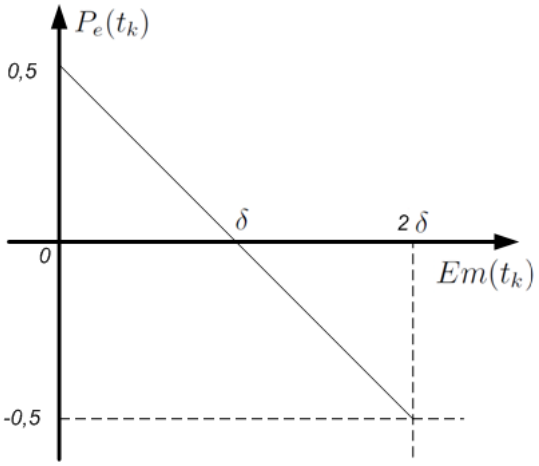

Figura 4.17: Função do controle Proporcional. Retorno positivo quando o erro médio é menor do que a tolerância e vice-versa

Diante deste comportamento, fez-se uso da Equação (4.23) para o ajuste do intervalo entre mensagens de sincronismo:

$$T_i(t_k) = T_i(t_{k-1}) * (1 + P(t_k)) \tag{4.23}$$

Como destacado anteriormente, o grande problema desta abordagem está no fato de que o intervalo entre mensagens de sincronismo está diretamente relacionado ao período entre duas atuações consecutivas do controlador. Em particular, poderá ocorrer a situação em que a sincronização comece a degradar, mas o controlador terá que esperar a próxima oportunidade para atuar. Este retardo poderá ser impeditivo para se alcançar a sincronização desejada. Sendo assim, é necessário um mecanismo que limite a correção do intervalo entre mensagens de sincronismo para evitar estas situações. No caso deste trabalho, usou-se a função derivadora (equação 4.21) para se obter a taxa de variação de $Em(t_k)$ e estimar o tempo necessário (ΔT) para que a sincronização chegue ao nível desejado. Pode-se resolver este problema com a seguinte igualdade:

$$Em(t_k) + \Delta T * D_e(t_k) = Em(t_k + \Delta T) = \delta \tag{4.24}$$

Ao se considerar $Em(t_k + \Delta T)$ como o valor da tolerância desejada, tem-se ΔT

será o limite de tempo para que erro observado não ultrapasse a margem permitida. Desta forma, pode-se definir, a cada oportunidade do ajuste de intervalo de mensagens de sincronismo, um limite para este ajuste, da seguinte forma:

$$\Delta T = \frac{\delta - Em(t_k)}{D(t_k)} \tag{4.25}$$

Finalmente, em conformidade com a quarta consideração citada no início desta seção, se nada for feito, a Equação (4.23) forçará uma situação de imposição da taxa dos nós que observarem o maior erro médio local ($Em(t_k)$). Ao invés disso, o que se pretende é induzir uma maior atividade da vizinhança para que os nós entrem em consenso mais rapidamente e reduzam $Em(t_k)$. Para isso, realiza-se uma simples média aritmética entre as Equações (4.23) e (4.18)

Os algoritmos (5) e (6) resumem a computação efetuada pelos nós nos instantes de recebimento e transmissão de mensagens de sincronismo, respectivamente.

Algorithm 5 A cada momento que o nó i recebe uma mensagem de sincronismo do nó j

atualizar $\overline{T}_j(t_n)$
armazenar $e(t_n)$

Algorithm 6 Nó i transmite uma mensagem de sincronismo para todos os seus vizinhos

calcular $Em(t_k)$, $P(t_k)$, $I(t_k)$ e $D(t_k)$;
if $0.9 * \delta < I_e(t_k) < 1.1 * \delta$ **then**
  sincronização adequada
else
  $T_i(t_k) \leftarrow T_i(t_{k-1}) * (1 + P(t_k))$
  $\Delta T \leftarrow \frac{\delta - Em(t_k)}{D(t_k)}$
  if $T_i(t_k) > \Delta T$ **then**
    $T_i(t_k) \leftarrow \Delta T$
  end if
end if
calcular a média entre $T_i(t_k)$ e $\overline{T}_j(t_k)$

4.4.3 Validação

Para avaliar o mecanismo de ajuste dinâmico do intervalo entre transmissões de mensagens de sincronismo, foram consideradas, praticamente todas as condições apresentadas na Seção 4.3, exceto pelo erro de *skew*, que foi ajustado para 90 ppm.

O cenário inicial foi configurado com 100 nós na topologia em anel, o tempo de simulação foi ajustado para 200.000 s e a tolerância desejada para $10us$. Em seguida, relaxou-se este valor para $20us$ e $40us$. As Figuras 4.18, 4.19 e 4.20 ilustram os resultados, no que se refere ao erro médio local. Além disso, foi registrado o intervalo médio entre mensagens de sincronismo, calculado da seguinte forma:

$$Tm = \frac{\sum_{i=0}^{N-1} (T_i(t_k))}{N} \tag{4.26}$$

onde i é o índice dos nós da rede e N é o número total de nós.

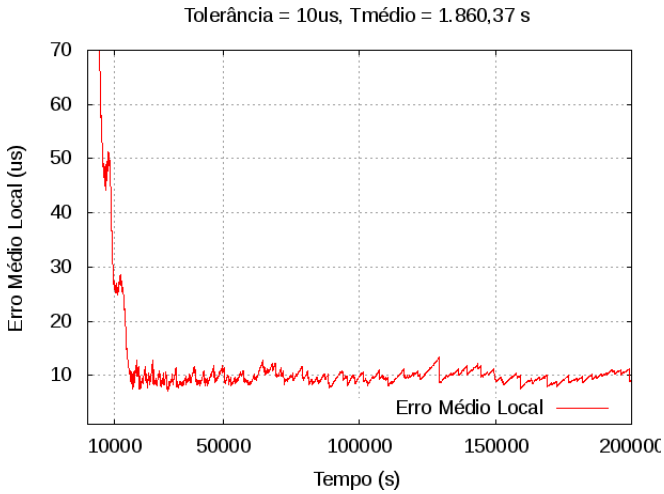

Figura 4.18: Ajuste dinâmico do intervalo de mensagens: anel de 100 nós, com 90 ppm de qualidade dos relógios e tolerância de $10us$.

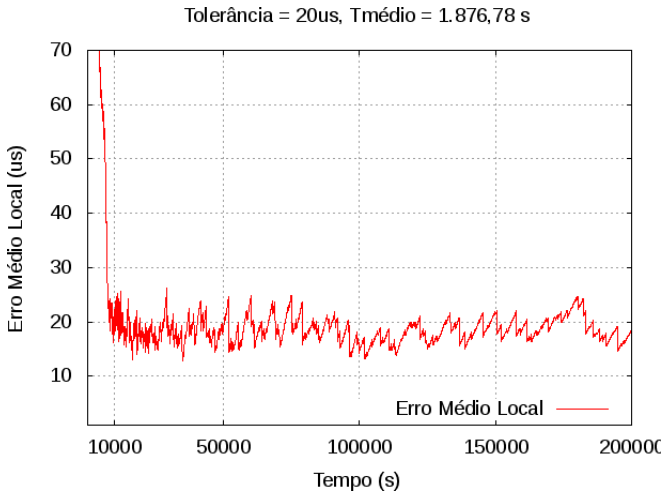

Figura 4.19: Ajuste dinâmico do intervalo de mensagens: anel de 100 nós, com 90 ppm de qualidade dos relógios e tolerância de $20us$.

A análise desses resultados para a topologia em anel deixa claro que a malha de controle está funcionando bem, uma vez que o erro médio local estabiliza em

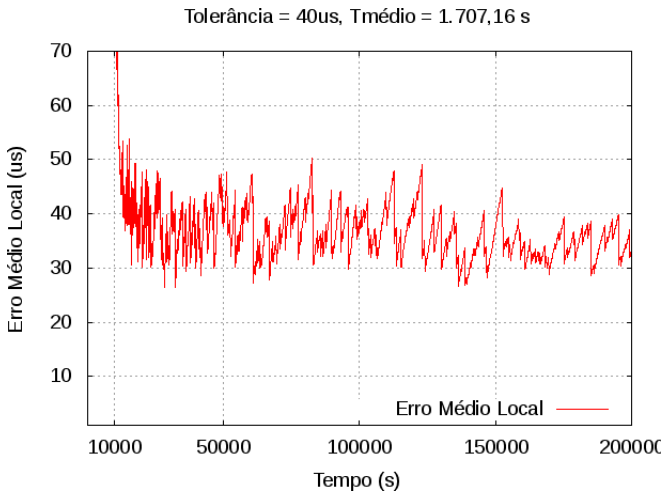

Figura 4.20: Ajuste dinâmico do intervalo de mensagens: anel de 100 nós, com 90 ppm de qualidade dos relógios e tolerância de $40us$.

torno da tolerância desejada. No entanto, um comportamento contra-intuitivo se apresenta: enquanto o intervalo médio entre mensagens de sincronismo fica na casa de $1860s$ para uma tolerância de $10us$, para tolerância de 40 us, este intervalo atinge o valor médio de $1707s$. Estas constatações parecem contrariar a premissa de que para uma melhor sincronização, maior tem que ser a atividade da rede e, consequentemente, menor tem que ser o período entre mensagens de sincronismo. Todavia, este comportamento é perfeitamente compatível com o modelo da malha de controle, que ao perceber que o erro está dentro de limites aceitáveis, não ajusta mais o intervalo entre mensagens (ver o algoritmo 6). Desta forma, o experimento ilustrado na Figura 4.18 alcança o seu ponto de estabilidade com um número maior de ajustes dos relógios em ralação ao experimento da Figura 4.20. Em outras palavras, o primeiro experimento, tolerância de $10us$, teve mais oportunidades para ajustar o seu intervalo entre mensagens de sincronismo, e provavelmente, isto levou a um melhor ajuste da taxa de progressão dos relógios lógicos. Esta abordagem sugere que a rede pode ter inúmeros pontos de operação estáveis, ou seja, para uma dada tolerância, a rede pode operar com vários intervalos entre mensagens de sincronismo, que irão depender do ajuste de taxa dos relógios lógicos. Intuitivamente, quanto melhor sincronizados estiverem os relógios lógicos, maior será o intervalo entre mensagens. Uma sugestão para que a malha de controle busque os pontos de operação com menor custo para a rede, seria permitir que a malha de controle ajuste os intervalos, apenas depois que o algoritmo atingisse o melhor nível de sincronização.

Por fim, realizou-se os mesmos experimentos, mudando-se apenas a topologia: GRID com 100 nós. Os resultados podem ser observados na Figura 4.21

Neste cenário também observou-se a convergência do erro médio local, porém os valores médios foram inferiores à tolerância designada. Em particular, observou-se para as tolerâncias de $10us$, $20us$ e $40us$, o erro médio local convergiu para $4.458us$,

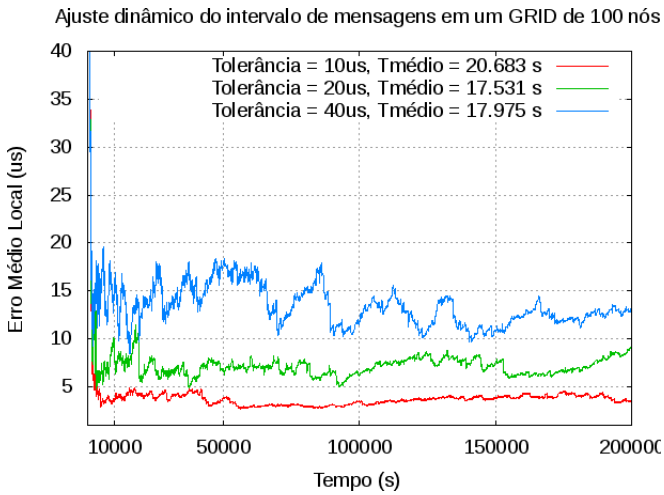

Figura 4.21: Ajuste dinâmico do intervalo de mensagens. GRID de 100 nós, com 90 ppm de qualidade dos relógios.

9.456us e 12.936us, respectivamente. Estes resultados sugerem que a malha de controle não está perfeitamente adequada às novas condições (rede mais densa e com diâmetro menor). Claramente, o ajuste das taxas de progressão dos relógios lógicos foi mais rápido do que o ajuste do intervalo de mensagens de sincronismo. De uma forma genérica, as contantes de controlador PID deveriam tornar-se variáveis, a serem ajustadas conforme as condições da rede. No entanto, esta observação não compromete o objetivo principal da implementação deste mecanismo, uma vez que a rede iniciou a operação com um intervalo médio de 30s e ao final da simulação atingiu a casa dos 20.000s

Capítulo 5

Viabilidade e Avaliação da Integração Escalonamento-Sincronização

O algoritmo de sincronização de relógios RGCS foi descrito e avaliado no capítulo anterior sob condições de transmissão de mensagens independentes de qualquer escalonamento. No entanto, o objetivo deste trabalho é o desenvolvimento de algoritmos que venham a suportar o modelo de comunicação TDMA.

Pode-se afirmar que para o correto funcionamento do referido modelo é necessário o consenso em taxa e *offset* dos relógios dos nós sensores. Por outro lado, o mecanismo de sincronização é dependente da temporização das mensagens de sincronismo e tem que ser persistente para impedir que o erro entre os relógios lógicos cresça indefinidamente. Sendo assim, fica clara a interdependência entre a sincronização e o modelo TDMA, ou seja, se a sincronização não funcionar, a qualidade da comunicação degrada, por outro lado, se a comunicação falhar, a sincronização não será possível.

Este capítulo tem por objetivo, investigar o impacto da integração do algoritmo RGCS a uma infraestrutura TDMA, cujo escalonamento foi gerado pelo algoritmo $node^2 - Sched$, descrito no Capítulo 3. Alguns cuidados tiveram que ser tomados, a fim de viabilizar tal funcionamento. Estes detalhes estão discriminados na próxima seção.

5.1 Considerações sobre a integração escalonamento-sincronização

Para que o modelo de comunicação TDMA funcione corretamente é necessário que todos os nós da rede conheçam o tamanho do *frame* de *slots*, assim como a marcação

de início ou fim de cada *frame*. Um dos problemas que podem acontecer, é que apesar da sincronização adequada, os nós tenham iniciado o sequenciamento de seus *frames* em momentos distintos, ou seja, os *frames* estão desalinhados, como representado na Figura 5.1. Outro fator a ser considerado é se a comunicação TDMA começou antes da correta sincronização dos relógios. Nesta situação, as marcações dos *frames* (e até mesmo dos *slots*) poderão ocorrer em momentos distintos para cada nó da rede, em função do erro entre os relógios.

Tais considerações deixam clara a necessidade de um procedimento para alinhamento e manutenção dos *frames* da estrutura TDMA.

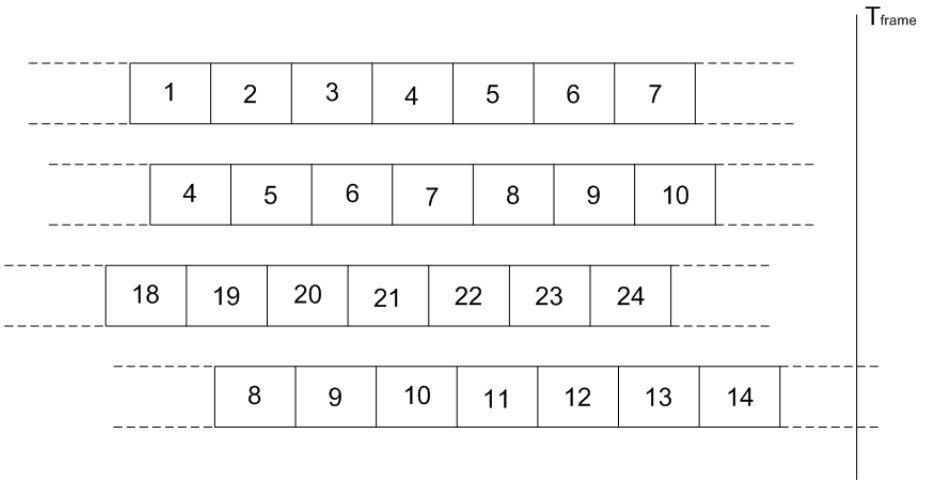

Figura 5.1: A sincronização não garante o alinhamento dos *frames*.

Para garantir o bom funcionamento da estrutura de escalonamento, é necessário que os nós alinhem seus *frames* de acordo com uma referência de tempo dos relógios lógicos, os quais, pela própria implementação do sincronizador, já estão em fase. Sendo assim, segue-se o seguinte raciocínio:

1. considera-se que o algoritmo $node^2 - Sched$ já operou na rede, fazendo com que todos os nós tenham conhecimento da referência dos slots disponíveis para a comunicação;

2. considera-se que algum mecanismo de disseminação levou a informação necessária a todos os nós da rede, para o cálculo do tamanho do *frame* (número de *slots*);

3. Cada nó entra em operação e inicia a contagem do seu relógio lógico. De posse do tamanho do *frame*, calcula o tempo lógico em que o *frame* terminará da seguinte forma:

$$L_f = n_s * t_s \tag{5.1}$$

onde L_f é o valor do relógio lógico no qual o frame terminará, n_s é o número de *slots* do *frame* e t_s é a duração de cada *slot*. Utilizando *timers* internos,

cada nó escalona dois eventos: o fim de *frame* e o início do primeiro *slot*, aguardando um pequeno período de segurança ($L_{s0} = diffs + t_s$);

4. o *timer* de escalonamento de *slots* é recursivo até o penúltimo *slot*. Sendo assim, a cada estouro deste *timer* é escalonado um novo *slot*. Mais ainda, é re-escalonado (corrigido) um novo fim de frame com relação ao valor do relógio atual do relógio lógico, ou seja:

$$L_f = L(t) + n_{sr} * t_s \tag{5.2}$$

onde $L(t)$ é no instante do re-escalonamento e n_{sr} corresponde ao número de *slots* que faltam para terminar o *frame*. Tal procedimento tem por objetivo alinhar e manter o alinhamento dos *frames*, ao custo inicial de se perder o acesso a alguns *slots*. Este efeito indesejado é causado pelo ajuste de *offset* dos relógios lógicos que podem abreviar o fim do *frame*.

5. a cada fim de frame ($L_f + diffs$), reinicia-se a contagem dos identificadores dos *slots* e procedimento é retomado. Considerando a adequada sincronização dos relógios lógicos dos nós sensores, esta estratégia mantém o alinhamento dos *frames* continuamente.

O mecanismo descrito anteriormente esbarra em um detalhe técnico: a base de tempo dos *timers* de um nó sensor é o relógio de hardware e não o relógio lógico. Como os relógios de hardware estão completamente defasados e progridem a taxas diferentes, a configuração destes *timers* não pode ser realizada com os valores dos relógios lógicos. Na verdade para que este esquema funcione apropriadamente é necessário a conversão do tempo do relógio lógico no tempo do relógio de hardware. Para tal, consideram-se as seguintes definições:

a) o relógio de hardware é na verdade um contador de pulsos, gerados por um oscilador a cristal;

b) os relógios têm uma taxa de progressão nominal correspondente a um período constante entre pulsos, doravante denominado T_n;

c) em função das diferentes características físicas dos cristais, os relógios podem apresentar variações em suas taxas de progressão. Sendo assim, cada nó tem uma taxa de progressão verdadeira, correspondente a um período constante entre pulsos, doravante denominado T_v;

d) o tempo medido no relógio lógico é função do relógio de hardware, conforme a equação 4.2;

e) o tempo medido no relógio de hardware é um múltiplo inteiro de T_n, ou seja, a referência de tempo disponível para um nó sensor corresponde ao número de pulsos N_p contados em um determinado intervalo de tempo universal, multiplicado pelo período entre pulsos, T_n;

Diante do acima exposto, pode-se estabelecer a seguinte modelagem:

1. Seja o valor do relógio lógico de um nó, no tempo universal t_k, denotado por $L(t_k)$. Em conformidade com a Equação 4.2, a progressão deste relógio pode ser modelada da seguinte forma:

$$L(t_k) = L(t_{k-1}) + l(t_k) * N_p * T_n \tag{5.3}$$

onde l é o fator de correção da taxa de progressão do relógio lógico.

2. Seja $N_p(t_k)$ o número de pulsos registrados pelo relógio de hardware durante o intervalo de tempo lógico compreendido entre $L(t_{k-1})$ e $L(t_k)$, onde t_{k-1} e t_k são dois instantes do tempo universal. Trabalhando a Equação (5.3) tem-se que:

$$N_p(t_k) = \frac{\Delta L}{l(t_k) * T_n} \tag{5.4}$$

Na verdade $N_p(t_k)$ também corresponde a variação do tempo universal, ou seja:

$$N_p(t_k) = \frac{t_k - t_{k-1}}{T_v} \tag{5.5}$$

logo,

$$\frac{L(t_k) - L(t_{k-1})}{l(t_k) * T_n} = \frac{t_k - t_{k-1}}{T_v} \tag{5.6}$$

A igualdade da equação 5.6 será verdadeira caso corresponda ao modelo da evolução do relógio lógico. Trabalhando esta equação, tem-se:

$$L(t_k) - L(t_{k-1}) = (t_k - t_{k-1}) * l(t_k) * \frac{T_n}{T_v} \tag{5.7}$$

que por definição (equação 4.2) reflete exatamente como o relógio lógico progride.

É importante observar que, obrigatoriamente, o valor de $N_p(t_k)$ tem que ser inteiro. Caso esta premissa não se confirme, fica entendido que ΔL foi superior ao intervalo correspondente a um número inteiro de pulsos, contados pelo relógio de hardware, conforme ilustrado na Figura 5.2. Desta forma, se nada for feito, a cada atualização do relógio lógico, uma pequena fração de tempo será perdida e o erro acumulado ao longo da operação.

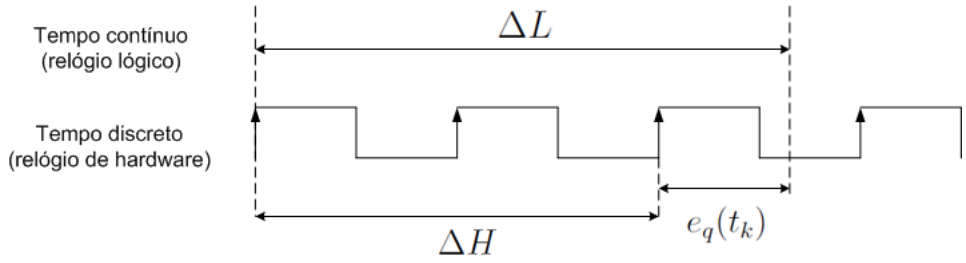

Figura 5.2: Erro de quantização: conversão do tempo lógico no tempo discreto.

3. Seja $e_q(t_k)$ o erro de quantização correspondente a parte fracionária de $N_p(t_k)$, discutido anteriormente e ilustrado na Figura 5.2. Pode-se obter este valor da seguinte forma:

$$e_q(t_k) = (N_p(t_k) - \lfloor N_p(t_k) \rfloor) * T_n \tag{5.8}$$

   Assim, a Equação 5.4 pode ser compensada para anular o erro de quantização da seguinte maneira:

$$N_p(t_k) = \frac{\Delta L + e_q(t_{k-1})}{l(t_k) * T_n} \tag{5.9}$$

   Esta abordagem garante o princípio da causalidade, não permitindo que o tempo universal retroceda.

4. Em termos práticos, escalonar um intervalo de tempo lógico (ΔL) em um nó sensor, significa escrever o valor $\lfloor N_p(t_k) \rfloor$ no *timer* do dispositivo.

5. No caso da simulação computacional, a operação do relógio de hardware é governada pelo relógio da simulação. No entanto, o relógio lógico continua funcionando com a base de tempo do relógio de hardware. Esta questão apesar de parecer complicada, pode ser facilmente superada, uma vez que na simulação, o período correspondente a taxa de progressão verdadeira do relógio de hardware (T_v) é conhecido. Sendo assim, escalonar um intervalo de tempo lógico (ΔL) em um nó sensor, significa escalonar $\lfloor N_p(t_k) \rfloor * T_v$ no relógio da simulação.

5.2 Estudo da viabilidade do RGCS

Com o intuito de ratificar a robustez do algoritmo RGCS, observada nas seções 4.3.2 e 4.3.3, resta implementá-lo em cenários mais realísticos, nos quais a sincronização é requisito fundamental para o cumprimento de tarefas. Dentre as aplicações nas quais se observam elevado grau de dependência da sincronização de relógios, uma das mais complexas é a comunicação TDMA, uma vez que este modelo de comunicação só é viável em função do consenso dos relógios em fase e em taxa de progressão. Por outro lado, como discutido no início deste capítulo, a manutenção do sincronismo

entre os relógios só é possível se houver êxito na comunicação entre os nós (troca de mensagens de sincronização).

O que se pretende nos próximos experimentos é avaliar a manutenção da sincronização, considerando uma infraestrutura TDMA livre de colisões. Neste cenário, os transceptores dos nós sensores permanecerão desligados durante todo o tempo não correspondente a um *slot* disponível para comunicação. Foram realizadas simulações para avaliar o erro médio local (EML), conforme a equação 4.16 em função do tamanho do frame (número de *time slots*) e da qualidade da sincronização no início da operação TDMA. Esta qualidade foi ajustada em função do número de mensagens de sincronismo trocadas antes do início do funcionamento do modelo de comunicação TDMA. No primeiro experimento, os cenários foram configurados de seguinte maneira:

- a modelagem dos relógios, o rádio escolhido e seus parâmetros foram os mesmos apresentados na seção 4.3;
- inicialmente foram usados apenas dois nós, posicionados a uma distância de 10 metros, o que garante a conexão rádio, considerados os parâmetros ajustados;
- a oportunidade de comunicação será sempre no *slot* 0. Sendo assim, aumentando-se o tamanho do *frame*, aumenta-se também o intervalo de transmissão das mensagens de sincronismo;
- o número de *time slots* variou da seguinte maneira: 1000, 10000, 50000, 100000, 200000, 300000, 400000, 500000;
- variou-se a imprecisão dos relógios de hardware da seguinte forma: 30, 60 e 90 ppm;
- a escolha da duração de cada *time slot* considerou os tempos de transição de estado do rádio CC2420, cujo retardo dominante corresponde à transição do estado **desligado** para **transmitindo** ou **recebendo**, no valor de $0,194ms$. Este retardo é conhecido como *wake-up time*, doravante denominado wk. Sendo assim, um pacote só pode ser transmitido depois de transcorrido um período mínimo de wk unidades de tempo, caso contrário o rádio do receptor ainda estará desligado e o pacote será perdido;
- seja t_g o período de tempo entre o início de cada *slot* de transmissão e o início da transmissão propriamente dita. Considerou-se $t_g = 1,5 * wk$;
- seja t_x o intervalo de tempo associado à transmissão de uma mensagem de sincronismo. Seja t_s a duração de um *time slot*. O valor de t_s é obtido da

seguinte forma:

$$t_s \geq 2 * t_g + t_x \tag{5.10}$$

Nos experimentos realizados, o valor de t_s foi o menor possível, ou seja, considerou-se a igualdade da inequação 5.10. A Figura 5.3 ilustra os tempos designados e auxilia o entendimento;

- antes da operação TDMA, os nós iniciam as transmissões de suas mensagens de sincronismo em instantes aleatórios, escolhidos a partir de uma distribuição uniforme correspondente aos primeiros $30s$ de atividade. Estas mensagens seguirão sendo transmitidas, periodicamente, a cada $30s$, até atingirem o número de transmissões configurado. Em seguida entram no modo TDMA;

- variou-se o número de transmissões de mensagens de sincronismo, antes do funcionamento TDMA, da seguinte maneira: 3, 5, 7, 10, 20.

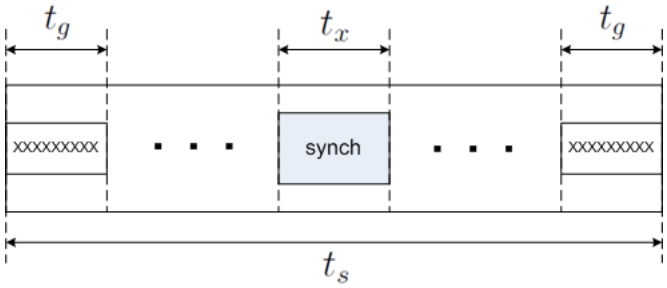

Figura 5.3: Temporização dos *time slots*. Necessidade do tempo de guarda (t_g) para compensar o *wake-up time*.

A investigação em questão, constatou que os resultados são bastante dependentes da condição inicial de sincronização da rede. Como citado anteriormente, tal condição diz respeito ao número de trocas de mensagens de sincronismo ocorridas antes da operação TDMA. Em outras palavras, para que o modelo TDMA integrado com a manutenção da sincronização dê certo, é necessário que a rede só comece a trabalhar com a infraestrutura de escalonamento (*slots* e *frames*), depois de atingir um nível mínimo de sincronização entre os nós. Sendo assim, as avaliações que se seguem consideraram cinco cenários distintos, nos quais, em cada um, estabeleceu-se o número de trocas de mensagens de sincronismo antes da operação TDMA. Os resultados podem ser observados na Figuras 5.4, 5.5 e 5.6, cada uma considerando os cinco cenários de condição inicial de sincronização e uma determinada qualidade para o relógio de hardware.

Pode-se inferir que sem uma sincronização inicial adequada, os *frames* desalinham e as mensagens de sincronismo são perdidas. Em consequência os relógios divergem indefinidamente. Em particular, pode-se observar nas Figuras 5.4 e 5.5

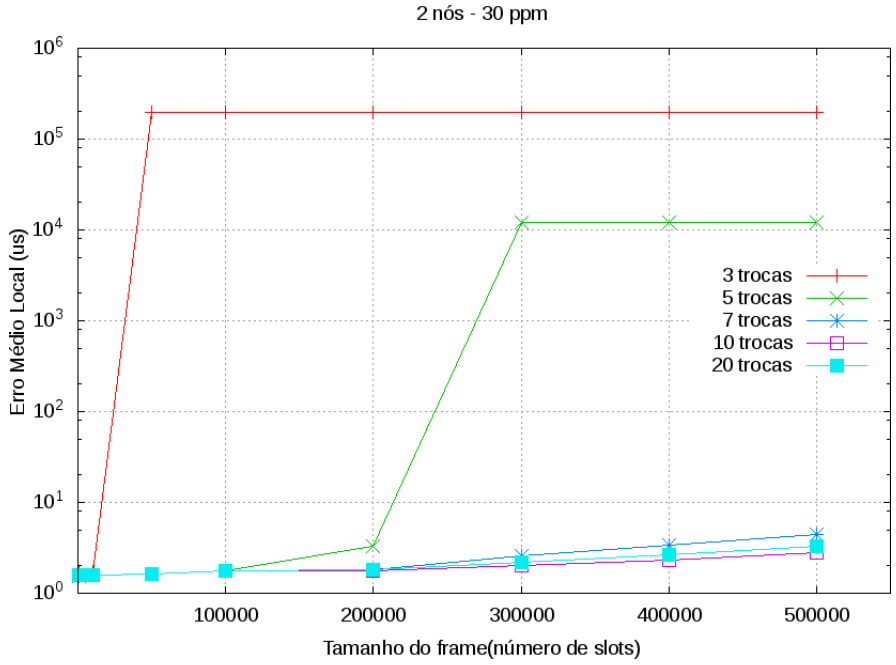

Figura 5.4: Funcionamento TDMA com dois nós. Imprecisão dos relógios de hardware de 30 ppm

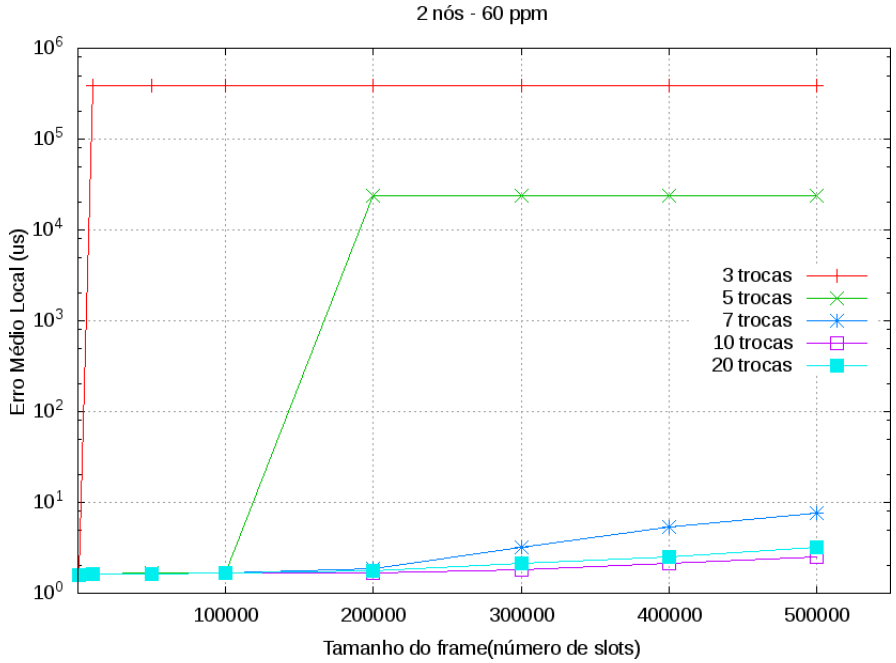

Figura 5.5: Funcionamento TDMA com dois nós. Imprecisão dos relógios de hardware de 60 ppm

que a partir de sete trocas de mensagens de sincronismo, o sistema se mantém funcionando com *frames* de até 500000 *slots*, considerando os relógios com qualidades de 30 e 60 ppm, respectivamente. Estes dois gráficos diferem, principalmente, em relação a curva com cinco trocas de mensagens, que para 30 ppm permanece funcionando até 200000 *slots*, enquanto para 60 ppm o funcionamento só é possível até

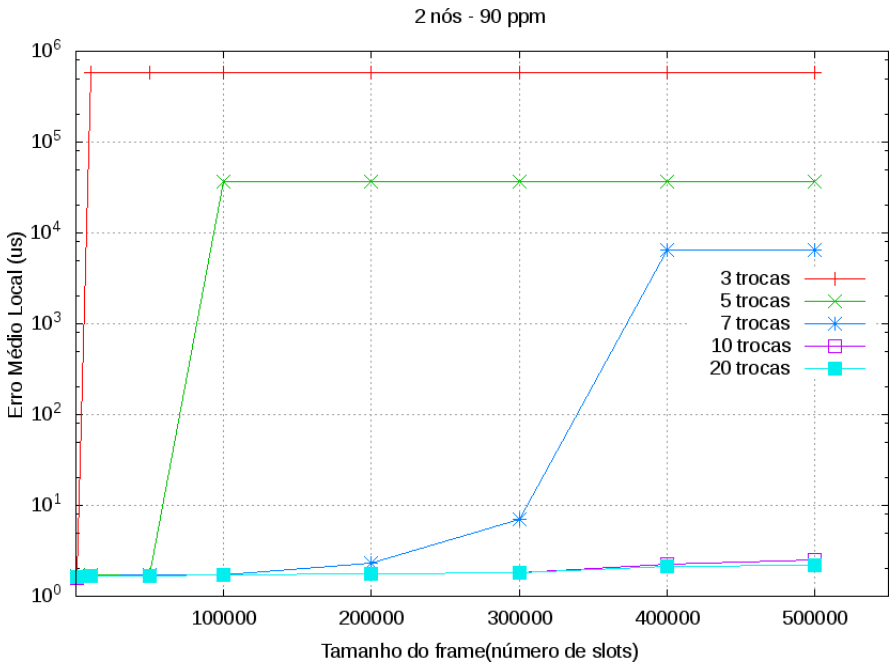

Figura 5.6: Funcionamento TDMA com dois nós. Imprecisão dos relógios de hardware de 90 ppm

metade desse valor. Para o cenário, no qual a variabilidade da taxa de progressão dos relógios é de 90 ppm (Figura 5.6) a sincronização só funciona com um *frame* de 500000 *slots* a partir de dez trocas de mensagens de sincronismo. Todos esses resultados são coerentes, no sentido que quanto pior a qualidade dos relógios, mais demorada será a sincronização dos mesmos. Ressalta-se, ainda, que as 10 mensagens de sincronismo trocadas para se obter um intervalo de manutenção da sincronização não inferior a 500000 *slots* representam um custo muito baixo. Os segmentos de reta horizontais, associados a elevados valores de erro médio local podem causar algum tipo de estranheza, pois induzem a um entendimento de que o erro médio local se estabiliza, ao invés de divergir indefinidamente com o aumento do tamanho do *frame*. Na verdade este erro é calculado considerando-se os últimos 1000 registros do erro médio local. Sendo assim, os valores associados aos referidos segmentos de reta dizem respeito ao valor médio dos últimos 1000 registros. Em outras palavras, o erro médio local associado a esses segmentos horizontais é dependente do tempo de simulação, que é o mesmo para todos os casos analisados.

Outra abordagem para se estudar o funcionamento da integração escalonamento-sincronização, foi observar redes com topologias no modelo grafo completo, ou seja, a cobertura rádio de cada nó alcança todos os outros nós da rede. As simulações foram feitas variando-se o número de nós da seguinte forma: 2, 4, 6, 8 e 10. Os resultados podem ser observados nas Figuras 5.7 e 5.8, cada um considerando 3 ou 5 trocas de mensagens, antes da operação TDMA, e ambos com a qualidade do relógio ajustada para 90 ppm.

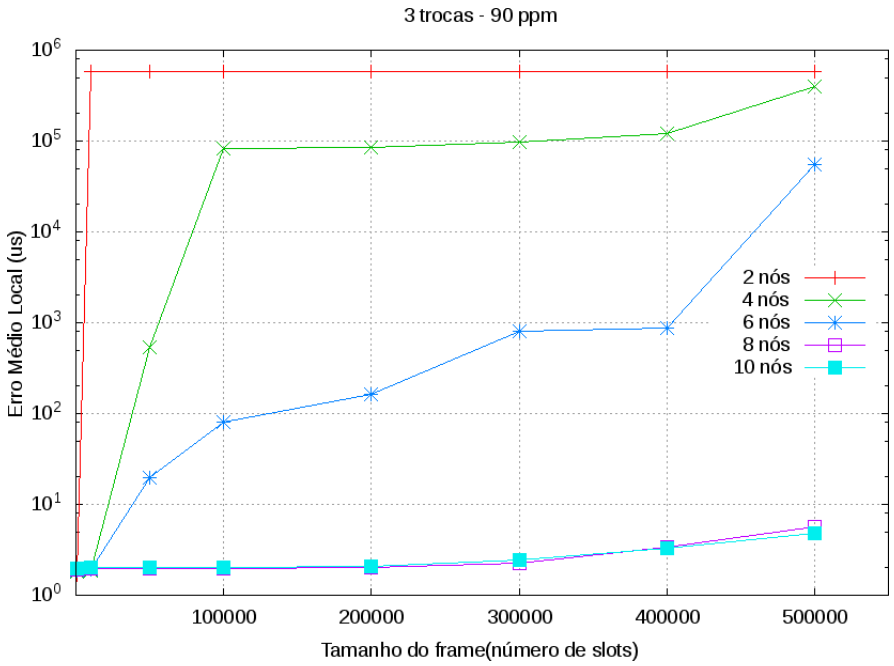

Figura 5.7: Funcionamento TDMA com grafos completos. Três trocas de mensagens e imprecisão dos relógios de hardware de 90 ppm

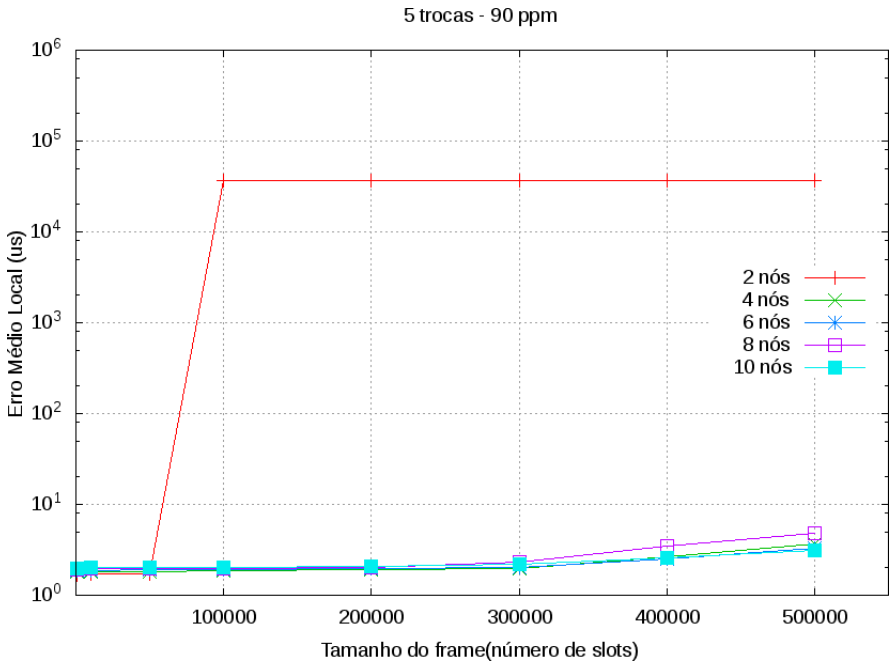

Figura 5.8: Funcionamento TDMA com grafos completos. Cinco trocas de mensagens e imprecisão dos relógios de hardware de 90 ppm

A análise individual das Figuras 5.7 e 5.8 confirma a intuição de que quanto maior for a densidade da rede, maior será a sua atividade e, consequentemente, maior será a convergência da sincronização. Destaca-se o comportamento das redes com 8 e 10 nós, as quais, com apenas 3 trocas iniciais de mensagens de sincronismo para cada par de nós, conseguem manter o sistema funcionando com *frames* de até 500000 *slots*. Considerando a Equação (5.10), pode-se obter o equivalente intervalo

de transmissão de mensagens de sincronismo, referente ao tamanho do frame. A seguir, calcula-se este intervalo, relativo ao frame com 500000 *slots*. Para tal, sabe-se que o pacote de dados na camada física carrega 72 bytes e a taxa de transmissão do radio utilizado é de 250000 bps.

$$t_x = \frac{72 * 8}{250000} = 0,02304(s) \tag{5.11}$$

Assim,

$$t_s = 2 * 1,5 * 0.000194 + 0,02304 = 0,02886(s) \tag{5.12}$$

Diante do resultado obtido na Equação (5.12), pode-se facilmente converter o tamanho do frame em tempo, que no experimento realizado, também corresponde ao intervalo entre mensagens de sincronismo. No caso de 500000 *slots*, este intervalo chega a 1443 s, o que ratifica a denominação do algoritmo em relação a sua robustez. Na verdade, estes experimentos confirmam o que já havia sido observado na Seção 4.4, na qual se buscava o máximo intervalo entre mensagens de sincronismo que atendesse a uma determinada tolerância.

A análise comparativa ratifica a ideia de que quanto mais sincronizada estiver a rede antes da operação TDMA, mas robusto será o esquema de manutenção. Salienta-se que com 5 trocas de mensagens de sincronismo, com exceção do cenário com dois nós, todos os outros mantiveram o sistema funcionando até o máximo tamanho de frame considerado (ver Figura 5.8).

5.3 A Integração

5.3.1 Escalonamento

Os algoritmos para escalonamento do tempo, desenvolvidos neste trabalho (Capítulo 3) se mostraram eficientes para a prover a informação acerca dos *time slots* disponíveis para nó ter acesso ao canal. Em outras palavras, estes algoritmos proveem a todos os nós da rede o rótulo dos *slots* utilizados para acesso ao canal, dentro de um determinado *frame*.

Neste trabalho considerou-se o algoritmo $node^2 - Sched$ para prover o referido escalonamento. Como descrito no Capítulo 3, o algoritmo estabelece uma coloração de nós à distância-2 em um grafo, no qual os nós representam os sensores e as arestas os enlaces de comunicação sem fio entre os nós. Associando-se a cada aresta o par ordenado das cores dos nós das extremidades de cada aresta. Este pares são organizados no formato (prefixo, sufixo), onde o prefixo será sempre a menor cor. As arestas passam a também estar coloridas à distância-2. As cores das arestas podem então ser associadas aos *time slots* correspondentes a cada enlace de comunicação,

considerando que nós vizinhos utilizam o canal alternadamente. Com a propriedade imposta pela coloração (distância-2), o uso do canal fica disciplinado de forma a impedir as colisões e, mais importante ainda, viabilizar o uso racional do sistema de comunicação, ou seja, permite que os rádios sejam desligados nos períodos ociosos.

Esta informação ainda não é suficiente para que um nó saiba o posicionamento de seus *slots* dentro de um determinado *frame*. Mais ainda, considerando que a referência aos *slots* (canal) é obtida a partir de uma estrutura circular, ou seja, a cada final de *frame*, reinicia-se a contagem das posições dos *slots*, é evidente que o conhecimento do tamanho do frame é essencial para que tal mecanismo funcione.

Para que cada nó da rede calcule o tamanho do *frame* a ser utilizado, bastaria o conhecimento das máximas ocorrências de um sufixo para cada prefixo. Exemplificando, se a máxima ocorrência do prefixo 3 é o sufixo 7 e sabendo que o sufixo é sempre maior que o prefixo, pode-se concluir que os pares (3 , 4), (3 , 5), (3 , 6) e (3 , 7) representam o número máximo de cores utilizadas. Assim, para cada máxima ocorrência o número de *slots* (ns_p) associados a um prefixo pode ser calculado da seguinte forma:

$$ns_p = s^p_{max} - p \tag{5.13}$$

onde s^p_{max} é a máxima ocorrência de um sufixo para o prefixo p. Assim o tamanho do frame seria facilmente calculado como a totalização dos ns_p, ou seja:

$$Ns = \sum_{p=0}^{n-1} ns_p \tag{5.14}$$

onde Ns é o número total de *slots* do frame e n é o número de cores utilizadas para coloração dos vértices.

Considerando a Figura 5.9, as máximas ocorrências são (0 , 7), (1 , 5), (2 , 7), (3 , 4), (4 , 7), (5 , 7) e (6 , 7). A totalização é facilmente obtida da seguinte maneira:

$$Ns = 7 + 4 + 5 + 1 + 3 + 2 + 1 = 23 \tag{5.15}$$

Ressalta-se que esta abordagem pode estimar um valor para Ns maior do que o mínimo necessário, pois na totalização são considerados todos os pares (prefixo, sufixo) possíveis. Analisando-se a Figura 5.9, pode-se notar que os pares (0 , 5) e (0 , 6) não foram utilizados.

Este trabalho considera que após a operação do algoritmo $node^2 - Sched$, as máximas ocorrências dos pares (prefixo, sufixo) são difundidas por um mecanismo qualquer, permitindo que todo o nó da rede calcule o tamanho do *frame*.

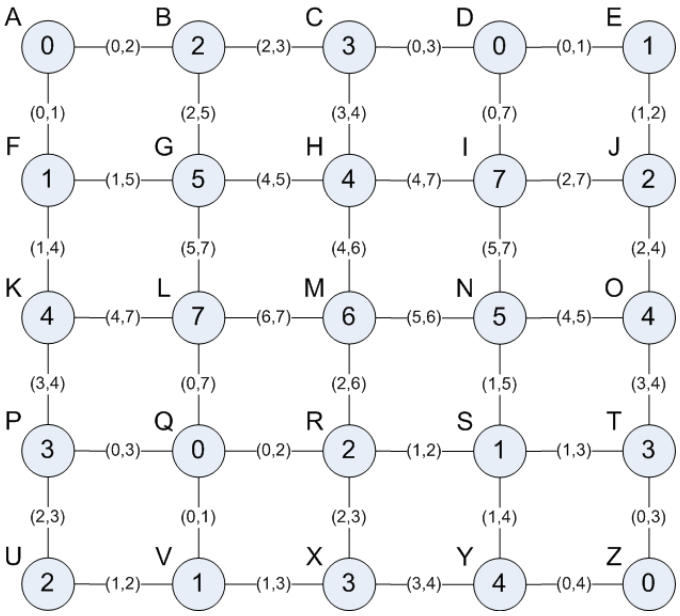

Figura 5.9: Coloração de arestas via coloração de vértices

5.4 A avaliação

Com o objetivo de avaliar um esquema TDMA, suportado pelos algoritmos desenvolvidos neste trabalho, criou-se um cenário de avaliação para as métricas de consumo de energia e desempenho da comunicação de dados. Para se obter um estudo comparativo, implementou-se um protocolo MAC CSMA/CD com possibilidade de configuração de um ciclo de trabalho, ou seja, de se estabelecer períodos em que o transceptor do nó sensor fique no modo SLEEP. A seguir, a descrição deste cenário:

a) **Topologia:** 25 nós dispostos em uma grade de 5x5. Considerando as linhas da grade, os pares de nós estão distantes 10m. Ver Figura 5.10;

b) **Escalonamento:** os nós da rede estão coloridos a distância-2, utilizando-se o algoritmo descrito na seção 3.2. Todos os nós sabem o tamanho do *frame* e posicionamento de seus *slots*, dentro do mesmo.

c) **Dados sensoreados:** os nós ***A***, ***E***, ***U*** e ***Z***, destacados com a cor azul claro na Figura 5.10, medirão temperatura na faixa de 4^oC a 45.5^oC e encaminharão tais medidas pela rede. Os nós iniciarão as medidas a partir de 1000s e repetirão o procedimento a cada período de pT (s). Este período é configurável, podendo variar entre os seguintes valores: $pT = 0.05,\ 0.1,\ 0.35,\ 0.5,\ 1,\ 5,\ 10,\ 20$. É importante reparar que quanto menor o pT maior a carga da rede e que a razão entre os limites superior e inferior desta faixa é igual a 400.

d) **Coleta dos dados sensoreados:** o nó ***M*** será o *sink* da rede, ou seja, será o destino final de todos os pacotes encaminhados.

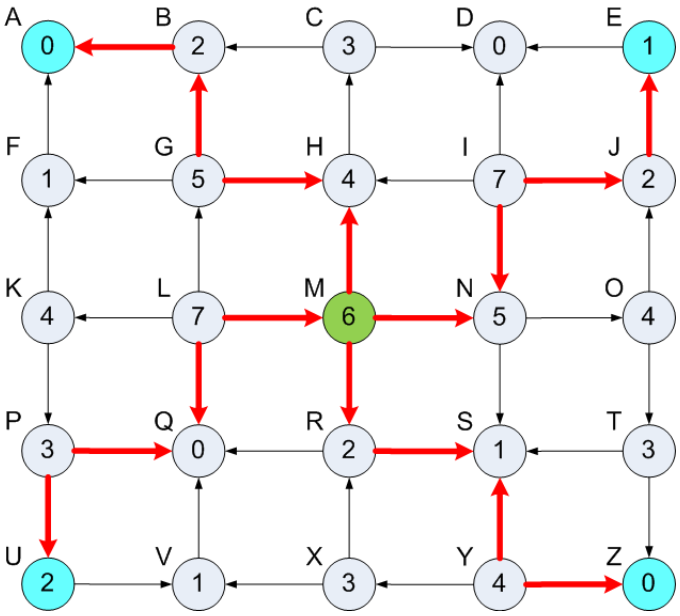

Figura 5.10: Cenário para avaliação dos esquemas TDMA e CSMA. Os nós ***A***, ***E***, ***U*** e ***Z*** realizam medições de temperatura periodicamente e as encaminham, pelas rotas destacadas na cor vermelha, até o nó ***M***

e) **Roteamento:** considera-se que a camada de aplicação teve acesso às rotas disponíveis para o escoamento dos dados, as quais estão discriminadas abaixo, juntamente com a correspondente sequencia de *slots*:

   1: $A - B - G - H - M \Longrightarrow 1 - 13 - 17 - 18$

   2: $E - J - I - N - M \Longrightarrow 7 - 15 - 21 - 20$

   3: $U - P - Q - L - M \Longrightarrow 11 - 2 - 6 - 22$

   4: $Z - Y - S - R - M \Longrightarrow 3 - 9 - 7 - 14$

   Cada segmento das 4 rotas disponíveis corresponde a um *time slot* atribuído na fase de escalonamento. Desta forma, antes de encaminhar um pacote, cada nó verifica a rota a ser utilizada, descobre o segmento a ser percorrido e consequentemente o *time slot* a ser utilizado para transmissão. Este procedimento é extremamente relevante, pois elimina a necessidade de transmissão de pacotes para vizinhos não inclusos na rota.

f) **Funcionamento:** cada nó entra em operação em um tempo aleatório, escolhido uniformemente dentro de um período inicial de 30s. Até os 1000s o tráfego da rede limita-se apenas as mensagens de sincronismo de relógio, escalonadas a cada 30s conforme descrição da Seção 4.3. A partir dos 1000s, as medições começam a ser realizadas pelos nós anteriormente definidos. A cada medição realizada, é criado um pacote específico com o dado medido e a rota a ser seguida. O pacote é armazenado em uma estrutura de dados FIFO (*buffer*), com dimensão de 10

Par ordenado	time slot		Par ordenado	time slot
(0 , 1)	0		(2 , 4)	12
(0 , 2)	1		(2 , 5)	13
(0 , 3)	2		(2 , 6)	14
(0 , 4)	3		(2 , 7)	15
(0 , 5)	4		(3 , 4)	16
(0 , 6)	5		(4 , 5)	17
(0 , 7)	6		(4 , 6)	18
(1 , 2)	7		(4 , 7)	19
(1 , 3)	8		(5 , 6)	20
(1 , 4)	9		(5 , 7)	21
(1 , 5)	10		(6 , 7)	22
(2 , 3)	11			

Tabela 5.1: Associação das cores dos nós aos *time slots*

posições. Para que um pacote deste *buffer* seja transmitido é necessário que sejam atendidas duas condições:

- o *time slot* atual não esteja sendo usado para transmissão de uma mensagem de sincronismo;
- o *time slot* atual corresponda a um segmento da rota daquele pacote.

Ao receberem pacotes com dados sensoreados os nós efetuam o mesmo procedimento descrito anteriormente, ou seja, armazenam o pacote recebido e o transmitem, de acordo com as condições estabelecidas acima. Em 2500s as medições cessam e a simulação termina aos 3000s.

O *sink* (nó $\boldsymbol{M}$) contabilizará os pacotes recebidos de cada fonte.

Com a totalização dos pacotes recebidos pelo nó *sink*, será possível avaliar o desempenho dos sistema de comunicação, ou seja, verificar a razão entre o número de pacotes recebidos pelo sink e o número de pacotes transmitidos pelas fontes. Outra métrica a ser avaliada será o consumo de energia dispendido pelos nós para os referidos sensoreamentos e comunicação de dados. Para efeito de comparação, como alertado anteriormente, usou-se um protocolo CSMA/CD para funcionar no lugar do esquema TDMA, segundo as mesmas condições. Inicialmente, este protocolo funcionou sem estabelecimento de qualquer ciclo de trabalho, ou seja, com os rádios ligados o tempo todo. Para tal, foram configurados os seguintes parâmetros:

- $randomTxOffset$(rTO) = 5 ms; este parâmetro define o intervalo de tempo $[0, rTO]$ do qual será, aleatoriamente, escolhido o retardo para transmissão do pacote. Este parâmetro é de suma importância para se evitar colisões.
- $backoffBasValue$(bBV) = 16 ms; tempo a ser aguardado para uma nova verificação do canal, caso o mesmo esteja sendo usado.

Ressalta-se que, propositalmente, a perda de pacotes não é tratada no protocolo CSMA/CD, de forma a poder avaliar as duas abordagens em igualdade de condições. A seguir os resultados dos experimentos:

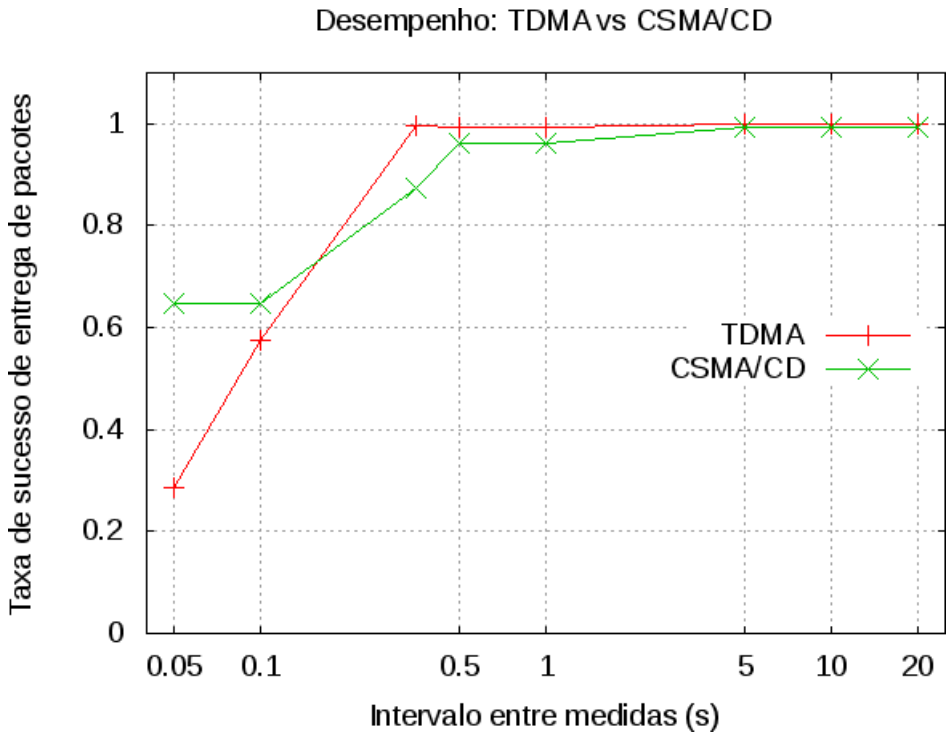

Figura 5.11: TDMA vs CSMA: comparação de desempenho da comunicação de dados para o cenário ilustrado pela Figura 5.10

Conforme observado na Figura 5.11, a partir de 0.35s de período entre medições, o esquema TDMA apresenta um desempenho de 100% na transmissão dos pacotes, ou seja, todas as medições chegaram ao nó *sink*. Abaixo deste valor, verifica-se a degradação do desempenho do protocolo, revelando-se aí, um limite para operação TDMA. Com relação ao esquema de contenção, verifica-se que a partir de 5s de período entre medições, o desempenho fica na casa de 99.3%. Antes disso, nos períodos de 0.5s e 1s o desempenho fica um pouco abaixo, num valor médio de 96.2%, e para o período de 0.35s, verifica-se 87.3% na taxa de sucesso de entrega de pacotes. Nos períodos que indicam as duas maiores frequências para a tomada de medidas, o desempenho do esquema CSMA/CD ficou em torno de 64.4%, denotando um melhor aproveitamento que o modelo TDMA.

De posse desta primeira comparação, pode-se agora, analisar o consumo de energia das duas abordagens, ilustrada na Figura 5.12 e concluir sobre os resultados apresentados. Considerando os 3000s de simulação, com medições entre 1000s e 2500s, o esquema TDMA teve um consumo médio de energia para cada nó de aproximadamente 91.8 *joules*, valor menor do que a metade do consumo apresentado no esquema CSMA, 203.6 *joules*. Ressalta-se que no esquema CSMA, a atividade

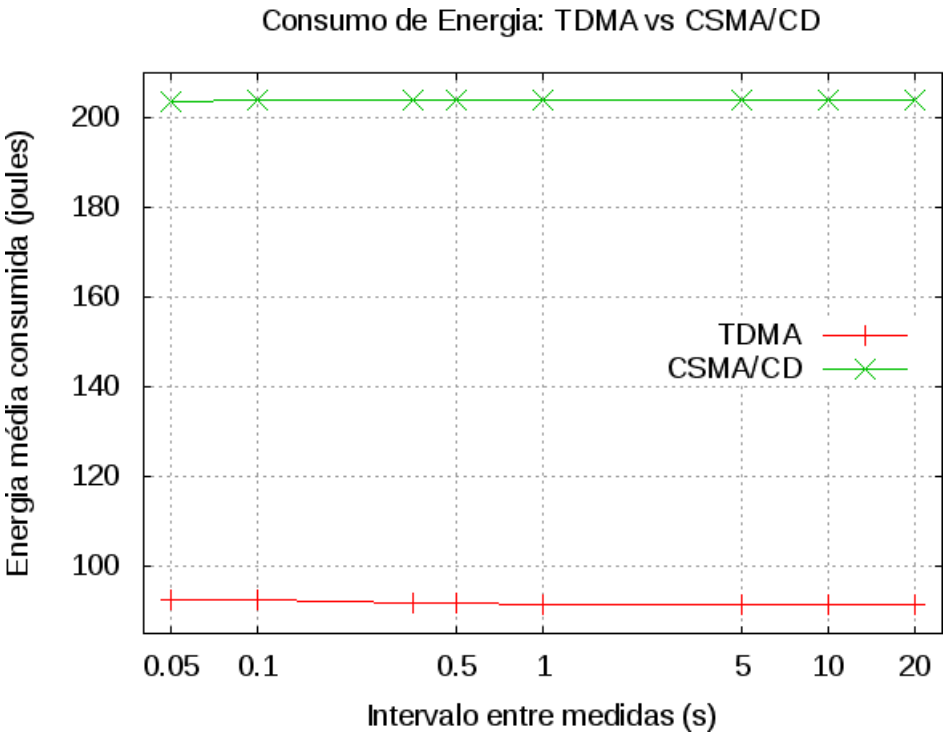

Figura 5.12: TDMA vs CSMA: comparação do consumo de energia (*joules*) em função do encaminhamento das medidas das fontes até o sink, conforme ilustrado pela Figura 5.10

da rede restringe-se ao período das medições (1500s), enquanto no funcionamento TDMA, a rede permanece o tempo todo em atividade para a manutenção do sincronismo.

Com o intuito de se investigar melhor o impacto do desligamento dos rádios no consumo de energia, considerou-se a avaliação do esquema CSMA/CD, com o estabelecimento de um ciclo de trabalho, cujo parâmetro a ser configurado é a fração do tempo que cada nó fica no modo SLEEP, fS. A este parâmetro foram atribuídos os seguinte valores: 0, 0.1, 0.3, 0.5, 0.7, 0.9. Adicionalmente, foram configurados os seguintes parâmetros:

- $listenInterval(lI) = 10ms$; este parâmetro define o tempo em que o rádio estará ligado. Juntamente com o parâmetro fS, pode-se então saber o período entre transição de estados..
- $beaconIntervalFraction(bIF) = 1$; este parâmetro define a duração dos *beacons* para acordar os vizinhos, como uma fração da duração do tempo em que cada nó fica no modo SLEEP. Sendo assim, com um valor menor que 1, pode-se considerar que, estatisticamente, bIF dos vizinhos de um nó serão acordados. No caso deste experimento, o valor configurado corresponde a 100%

Os resultados são apresentados a seguir:

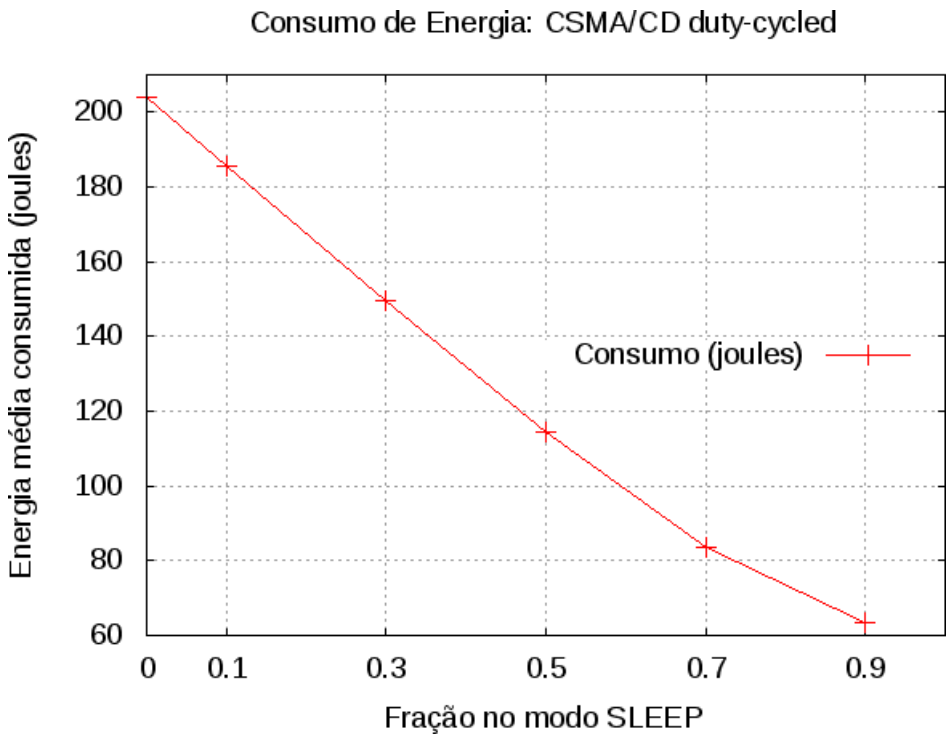

Figura 5.13: CSMA: consumo de energia, variando-se o ciclo de trabalho dos nós, conforme cenário ilustrado pela Figura 5.10

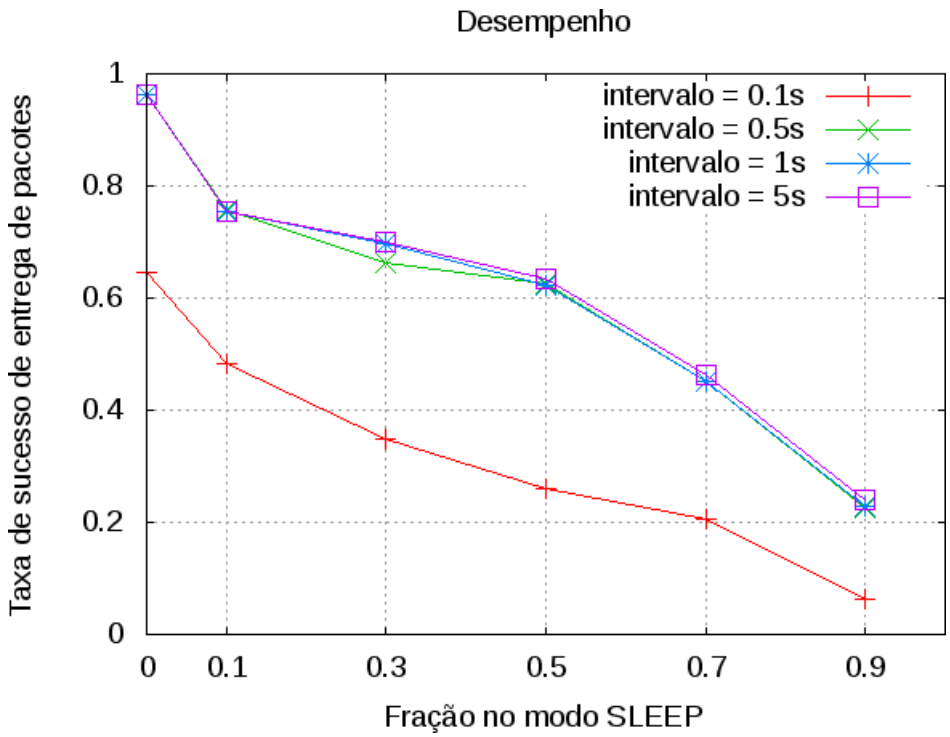

Figura 5.14: CSMA: desempenho, variando-se o ciclo de trabalho dos nós, conforme cenário ilustrado pela Figura 5.10

Analisando-se a Figura 5.13, pode-se chegar a conclusão que o mecanismo de estabelecimento de um ciclo de trabalho soluciona o problema do gasto de energia, uma vez que para frações no modo SLEEP de 0.7 e 0.9, o consumo fica abaixo

do observado para o modo TDMA (ver Figura 5.12). No entanto, a luz do que se pode observar na Figura 5.14, a taxa de sucesso na entrega de pacotes reduz significativamente com aumento da fração do tempo no modo SLEEP. Para os já mencionados valores 0.7 e 0.9, as taxas de sucesso são de 0.45 e 0.22, respectivamente. Cabe reparar que o consumo de energia não depende da carga da rede e sim da fração do ciclo de trabalho que os rádios estão desligados. Isto justifica o fato da Figura 5.13 apresentar uma única curva.

A partir dos resultados apresentados nesta seção, pode-se concluir que o esquema TDMA, com base nos algoritmos propostos neste trabalho, é viável, com resultados promissores em relação a minimização do consumo de energia e com desempenho adequado às demandas de grande parte das aplicações de RSSF.

Capítulo 6

Conclusão

A pesquisa bibliográfica, realizada neste trabalho, acerca dos desafios para a implementação de redes de sensores sem fio, revelou que a baixa oferta de energia para o funcionamento dos nós sensores é um dos principais limitadores da prospecção desta tecnologia. Ainda com base nesta pesquisa, identificou-se o sistema de comunicação sem fio como a principal fonte de consumo de energia, uma vez que os rádios utilizados nas aplicações desta área são transceptores especiais de curto alcance e com uma distinta característica: o consumo de energia transmitindo, recebendo ou em tempo ocioso é praticamente o mesmo. Estes transceptores só reduzem substancialmente seu consumo quando estão no estado *sleep*.

Tal constatação levou ao estudo dos protocolos de acesso ao meio, responsáveis pela coordenação do acesso ao canal e pelo controle dos transceptores. Uma abordagem para a comunicação eficiente consiste em dividir o tempo em *slots*, organizando-os em *frames*, de modo que os nós possam escalonar transmissões e recepções livres de colisões, *overhearing* e *overmitting*. Em períodos ociosos, os rádios podem ser simplesmente desligados. de maneira que os enlaces possam se estabelecer na temporização correta.

Face a complexidade apresentada, grande parte das soluções para este desafio baseou-se em uma referência central para a referida coordenação e/ou na formação de uma estrutura topológica rígida.

Neste trabalho foram desenvolvidos e avaliados dois algoritmos para escalonamento de enlaces e um algoritmo para sincronização de relógios.

Em relação ao escalonamento, um dos algoritmos, o $node^2 - Sched$, mostrou-se adequado às restrições impostas pelas RSSF, em função das seguintes razões:

- execução plenamente distribuída;
- não necessita de identificadores globais;
- simplicidade na implementação;

- baixo custo de mensagens, quando comparado a outra solução similar GANDHAM *et al.* (2008).

No que tange a sincronização, o algoritmo RGCS utilizou a propriedade gradiente, introduzida por FAN e LYNCH (2004), para estabelecer um mecanismo distribuído para ajuste (taxa e *offset*) dos relógios. A implementação e a avaliação deste algoritmo revelaram vários motivos que o fazem figurar como uma alternativa bastante relevante no cenário das RSSF. Dentre todos os motivos, aqueles de maior destaque são:

- execução plenamente distribuída sem a necessidade de um referência de tempo;
- operação independente de topologia;
- sem parâmetros de ajuste;
- sincronização efetuada a cada mensagem recebida;
- ajuste de taxa com elevada acurácia, permitindo um aumento considerável dos intervalos entre mensagens de sincronismo e a consequente economia de energia;
- baixos valores de erro médio local e global entre os relógios lógicos da rede, quando comparados aos resultados de outros trabalhos (RANGANATHAN e NYGARD (2010));
- elevada tolerância à perda de pacotes e à variabilidade dos intervalos entre mensagens de sincronismo.

Finalizando este trabalho, realizou-se um estudo de viabilidade dos algoritmos propostos, no sentido de avaliá-los em termos da correção do escalonamento livre de colisões e da manutenção da sincronização durante a operação TDMA. Os resultados mostraram que a taxa de perda de pacotes é nula sob determinado tráfego da rede, refletindo o êxito no escalonamento dos enlaces. No entanto, observou-se um limite para o bom funcionamento, claramente relacionado com a carga da rede.

A comparação com um modelo CSMA *duty-cycled* provou que a abordagem TDMA é a melhor opção considerando em conjunto as métricas de taxa de sucesso de transmissão de pacotes e consumo de energia.

6.1 Trabalhos Futuros

No decorrer deste trabalho, surgiram algumas questões que foram resolvidas pontualmente, sem que houvesse uma investigação mais profunda, ou mesmo um desenvolvimento adicional. Os temas discriminados a seguir representam solo fértil para

trabalhos de pesquisa futuros, e caso venham a produzir bons resultados, têm grande chance de agregar relevante valor ao trabalho desenvolvido nesta tese, podendo, até mesmo, vir a culminar em um produto de grande valor para as RSSF.

6.1.1 Desenvolvimento de um mecanismo para transição dos protocolos MAC

É de se esperar que após a instalação inicial dos nós sensores, estes iniciem a comunicação com base em um protocolo MAC de contenção (seção 2.1.1), uma vez que ainda não existe o conhecimento da topologia da rede e nem relógios sincronizados. Não haverá qualquer problema no uso deste protocolo para o estabelecimento da infraestrutura de escalonamento com o algoritmo $node^2 - Sched$ (seção 3.2), já que este é assíncrono e não depende da temporização das mensagens. Com relação a sincronização dos relógios, o desafio é bem mais complexo, pois as incertezas nos atrasos de transmissão das mensagens dominam os limites da precisão do sincronismo. Sendo assim, é de suma importância que se estude uma alternativa, de forma a adaptar o protocolo MAC inicial, para que a transmissão de mensagens de sincronismo, especificamente, tenham seu atraso dominado pela componente determinística.

Garantida a sincronização dos relógios, resta informar a rede o momento do chaveamento entre o protocolo legado e o modelo TDMA. Este é mais um desafio a ser solucionado.

6.1.2 Desenvolvimento de um algoritmo para difusão do tamanho do frame TDMA

Como alertado na seção 5.3.1, este trabalho não contemplou o desenvolvimento de um algoritmo para difundir as informações necessárias para se definir o tamanho do frame TDMA, assim como da posição relativa dos slots dentro do mesmo. Sabe-se, no entanto, que a organização dos rótulos dos *time slots*, conforme apresentada na seção 5.3.1 provê tais informações, bastando apenas que todos os nós da rede conheçam as máximas ocorrências de um sufixo, para cada prefixo. Em outras palavras, faz-se necessária a difusão dessa informação para que os nós da rede calculem o tamanho do frame e determinem a posição de seus *slots* associados dentro do mesmo.

Uma vez que algoritmo $node^2 - Sched$, além de prover os rótulos dos *slots* a serem utilizados por cada nó, também associa uma orientação às arestas do grafo de conexões, sugere-se o uso da técnica denominada *sink-decomposition* (GONÇALVES *et al.* (2010)) para se efetuar a difusão da informação citada. Para entender o funcionamento desta abordagem seguem algumas definições:

- Todo nó em que todas as suas arestas estão voltadas para si, é denominado *sink*;

- seja λ o valor da máxima distância, em número de saltos, de um nó da rede até um *sink*;

Dessas definições infere-se, que todo o nó em um grafo orientado tem um λ associado, em particular, os *sinks* têm $\lambda = 0$. Desta forma pode-se estratificar a rede segundo este parâmetro. A Figura 6.1 ilustra esta abordagem.

Com esta organização, fica estabelecido que somente os *sinks* têm a oportunidade de transmissão dos arranjos (P_i, S_{max}), onde P_i é a cor de um nó i e S^i_{max} é a maior cor de um vizinho do nó i. A cada mensagem dessa recebida, os nós registram ou atualizam os registros das máximas ocorrências. Após a transmissão, os *sinks* revertem suas arestas, passando o direito de transmissão para os novos *sinks* (λ imediatamente superior), que transmitirão os seus registros de máximas ocorrências. Este fluxo segue até o nível de λ máximo e retorna para os *sinks* originais. Nesta fase, os *sinks* estão de posse da informação de toda a rede e podem calcular o tamanho do frame e definir as posições dos *slots* associados. A partir daí, inicia-se um novo fluxo, na direção do nível de λ máximo para disseminação deste informação. Ao fim deste fluxo, toda a rede terá a informação necessária para a definição da infraestrutura TDMA.

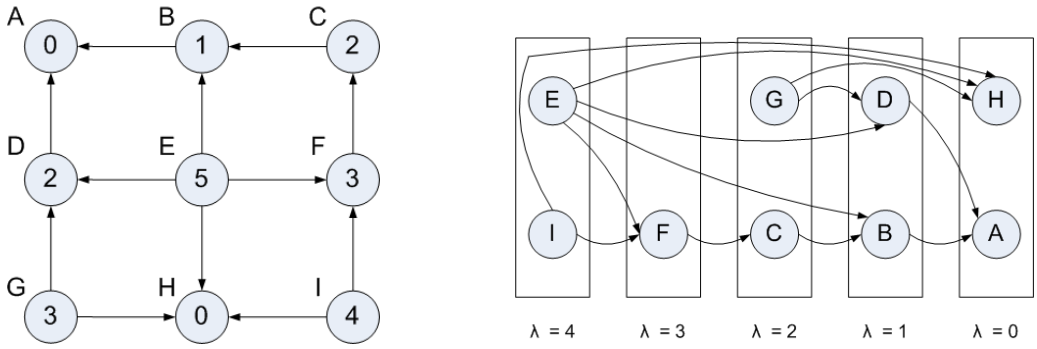

Figura 6.1: Exemplo da abordagem *sink decomposition*.

6.1.3 Estudo teórico, modelagem e desenvolvimento da malha de controle para ajuste dinâmico dos intervalos entre mensagens de sincronismo

A implementação do ajuste dinâmico do intervalo de mensagens de sincronismo, apresentada na seção 4.4 foi realizada com o objetivo principal de se provar que o mecanismo de manutenção do sincronismo entre relógios pode ser otimizado para re-

laxar consideravelmente tal atividade, desonerando a rede do consumo desnecessário de energia.

A abordagem implementada neste trabalho baseou-se em um controlador PID (seção 4.4.1) e conforme os resultados apresentados na seção 4.3, aumentou o intervalo entre mensagens de sincronismo por um fator de aproximadamente 600, o que justifica a adoção de tal procedimento. No entanto, dependendo da configuração inicial da rede, esta pode apresentar diferentes pontos de operação para atender ao requisito de sincronização desejado. Em outras palavras, a rede pode convergir para a sincronização desejada, usando diferentes intervalos para a transmissão de mensagens. Sendo assim, esta abordagem ainda carece de um estudo mais profundo, não só no contexto da modelagem da malha de controle, mas também acerca da escolha de seus parâmetros de configuração. No caso deste trabalho, os parâmetro foram os seguintes: faixa de retorno da função de monitoração do erro local médio, faixa do intervalo de confiança a ser monitorada pelo módulo integrador e coeficientes da médias móveis para as componentes integradora e derivadora.

6.1.4 Estudo de cenários dinâmicos: nós entrando e saindo

Considerando a abrangência das aplicações das RSSF, não é difícil imaginar situações em que os nós sensores deixam de funcionar por diversos motivos ou, novos nós entrem em funcionamento depois de que a rede já está em operação. Considerando que o modelo TDMA já está em funcionamento, surgem as seguintes questões:

- Qual o procedimento a ser adotado, por novos nós, para se anunciarem e manterem comunicação com a rede, antes de se adequarem ao modelo TDMA?
- Quais *slots* do *frame* TDMA o novo nó poderá usar?
- Como tratar as situações em que não há mais *slots* disponíveis para uso?
- Como reaproveitar *slots* de nós que pararam de funcionar?
- Qual será a política para sincronização de novos nós?

Todos esses questionamentos precisam ser cuidadosamente estudados para que as aplicações dessa nova proposta não sejam restritas a uma pequena quantidade de casos de uso. Estes eventos têm que ser cuidadosamente tratados para que não inviabilizem os avanços alcançados neste trabalho.

Referências Bibliográficas

AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., et al. "A survey on sensor networks", *Communications Magazine, IEEE*, v. 40, n. 8, pp. 102–114, 2002.

KAHN, J. M., KATZ, Y. H., PISTER, K. S. J. "Emerging Challenges: Mobile Networking for Smart Dust", *Journal of Communications and Networks*, v. 2, pp. 188–196, 2000.

SEAH, W. K. G., EU, Z. A., TAN, H.-P. "Wireless sensor networks powered by ambient energy harvesting (WSN-HEAP) - Survey and challenges", *2009 1st International Conference on Wireless Communication Vehicular Technology Information Theory and Aerospace Electronic Systems Technology*, pp. 1–5, 2009.

POTTIE, G. J., KAISER, W. J. "Wireless integrated network sensors", *Commun. ACM*, v. 43, pp. 51–58, May 2000. ISSN: 0001-0782.

YE, W., HEIDEMANN, J. *Medium Access Control in Wireless Sensor Networks*. Technical Report ISI-TR-580, USC/Information Sciences Institute, October 2003.

RUIZ, L. B., CORREIA, L. H. A., VIEIRA, L. F. M., et al. "Arquiteturas para Redes de Sensores Sem Fio", *sensornetdccufmgbr*, pp. 167–218, 2004.

DEMIRKOL, I., ERSOY, C., ALAGOZ, F. "MAC protocols for wireless sensor networks: a survey", *Communications Magazine, IEEE*, v. 44, n. 4, pp. 115–121, 2006. Survey.

REASON, J. M., RABAEY, J. M. "A study of energy consumption and reliability in a multi-hop sensor network", *SIGMOBILE Mob. Comput. Commun. Rev.*, v. 8, n. 1, pp. 84–97, 2004. ISSN: 1559-1662.

LANGENDOEN, K. "Preprint of a book chapter in ;"Medium Access Control in Wireless Networks, Volume II: Practice and Standards"; edited by H.Wu and Y. Pan", 2008.

LIU, S., SRIVASTAVA, R., KOKSAL, C. E., et al. "Achieving Energy Efficiency with Transmission Pushbacks in Sensor Networks". In: *IWQoS*, pp. 160–170, 2008.

YE, W., HEIDEMANN, J., ESTRIN, D. "An energy-efficient MAC protocol for wireless sensor networks". In: *INFOCOMM*, pp. 1567–1576 vol.3, 2002. Disponível em: <citeseer.ist.psu.edu/ye01energyefficient.html>.

VAN DAM, T., LANGENDOEN, K. "An adaptive energy-efficient MAC protocol for wireless sensor networks". In: *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 171–180, New York, NY, USA, 2003. ACM Press. ISBN: 1581137079.

POLASTRE, J., HILL, J., CULLER, D. "Versatile low power media access for wireless sensor networks". In: *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 95–107, New York, NY, USA, 2004. ACM Press. ISBN: 1581138792.

EL-HOIYDI, A., DECOTIGNIE, J. D. "WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks". In: *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, v. 1, pp. 244–251 Vol.1, 2004.

PINHO, A. C., SANTOS, A. A., FIGUEIREDO, D. R., et al. "Work in Progress: Two ID-Free Distributed Distance-2 Edge Coloring Algorithms for WSNs". In: *Networking 2009: Proceedings of the 8th event of the series of International Conferences on Networking*, pp. 919–930. IFIP International Federation for Information Processing - LNCS 5550, 2009.

FAN, R., LYNCH, N. "Gradient clock synchronization", *Proceedings of the twentythird annual ACM symposium on Principles of distributed computing PODC 04*, pp. 320–327, 2004.

PINHO, A. C., FIGUEIREDO, D. R., FRANÇA, F. M. G. "A Robust Gradient Clock Synchronization Algorithm for Wireless Sensor Networks". In: *COMSNETS 2012: Proceedings of the 4th event of the series of International Communications Systems an Networks*, 2012.

GARCÍA HERNANDO, A., MARTÍNEZ ORTEGA, J., LÓPEZ NAVARRO, J. "Problem Solving for Wireless Sensor Networks". first ed., Springer Science, 2009. ISBN: 978-1-84800-202-9.

HILL, J. L., CULLER, D. E. "Mica: A Wireless Platform for Deeply Embedded Networks", *IEEE Micro*, v. 22, n. 6, pp. 12–24, 2002. ISSN: 0272-1732.

YE, W., SILVA, F., HEIDEMANN, J. "Ultra-low duty cycle MAC with scheduled channel polling". In: *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pp. 321–334, New York, NY, USA, 2006. ACM. ISBN: 1-59593-343-3.

LU, G., KRISHNAMACHARI, B., RAGHAVENDRA, C. S. "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks", *Parallel and Distributed Processing Symposium, International*, v. 13, pp. 224a, 2004.

ERGEN, S. C., VARAIYA, P. "PEDAMACS: Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks", *IEEE Transactions on Mobile Computing*, v. 5, n. 7, pp. 920–930, 2006. ISSN: 1536-1233.

RAJENDRAN, V., OBRACZKA, K., GARCIA-LUNA-ACEVES, J. "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks", *Wireless Networks*, v. 12, n. 1, pp. 63–78, February 2006. ISSN: 1022-0038.

RAJENDRAN, V., GARCIA-LUNA-ACEVES, J., OBRACZKA, K. "Energy-Efficient, Application-Aware Medium Access for Sensor Networks". In: *2nd IEEE Conf. on Mobile Ad-hoc and Sensor Systems (MASS 2005)*, Washington, DC, nov. 2005.

VAN HOESEL, L. F. W., HAVINGA, P. J. M. "A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks: Reducing Preamble Transmissions and Transceiver State Switches". In: *1st International Workshop on Networked Sensing Systems (INSS, Tokio, Japan*, pp. 205–208, Tokio, Japan, 2004. Society of Instrument and Control Engineers (SICE). ISBN: 4-907764-21-9.

CHATTERJEA, S., VAN, L. H., HAVINGA, P. "AI-LMAC: An Adaptive, Informationcentric and Lightweight MAC Protocol for Wireless Sensor Networks". In: *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference*, pp. 381–388. IEEE, 2004.

SHI, L., FAPOJUWO, A. O. "TDMA Scheduling with Optimized Energy Efficiency and Minimum Delay in Clustered Wireless Sensor Networks", *IEEE Transactions on Mobile Computing*, v. 9, pp. 927–940, 2010. ISSN: 1536-1233.

RHEE, I., WARRIER, A., AIA, M., et al. "Z-MAC: a hybrid MAC for wireless sensor networks". In: *SenSys '05: Proceedings of the 3rd international*

conference on Embedded networked sensor systems, pp. 90–101, New York, NY, USA, 2005. ACM Press. ISBN: 159593054X.

ZHENG, T., RADHAKRISHNAN, S., SARANGAN, V. "PMAC: An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks", *Parallel and Distributed Processing Symposium, International*, v. 13, pp. 237a, 2005. ISSN: 1530-2075.

HALKES, G. P., LANGENDOEN, K. "Crankshaft: An Energy-Efficient MAC-Protocol for Dense Wireless Sensor Networks". In: *EWSN*, pp. 228–244, 2007.

KULKARNI, S. S., ARUMUGAM, M. "SS-TDMA: A Self-Stabilizing MAC for Sensor Networks". cap. In Sensor Network Operations, IEEE Press, 2006. in press.

ERGEN, S. C., VARAJA, P. *TDMA scheduling algorithms for sensor networks*. Relatório técnico, 2005. Disponível em: <`http://paleale.eecs.berkeley.edu/~varaiya/papers_ps.dir/tdmaschedule.pdf`>.

PANTAZIS, N. A., VERGADOS, D. J., VERGADOS, D. D., et al. "Energy efficiency in wireless sensor networks using sleep mode TDMA scheduling", *Ad Hoc Netw.*, v. 7, n. 2, pp. 322–343, 2009. ISSN: 1570-8705.

GANDHAM, S., DAWANDE, M., PRAKASH, R. "Link scheduling in wireless sensor networks: Distributed edge-coloring revisited", *J. Parallel Distrib. Comput.*, v. 68, n. 8, pp. 1122–1134, 2008.

RANGANATHAN, P., NYGARD, K. "TIME SYNCHRONIZATION IN WIRELESS SENSOR NETWORKS: A SURVEY". april 2010. Disponível em: <`http://www.airccse.org/journal/iju/papers/0410iju6.pdf`>.

ROMER, K., BLUM, P., MEIER, L. "Time Synchronization and Calibration in Wireless Sensor Networks". In: Stojmenovic, I. (Ed.), *Handbook of Sensor Networks: Algorithms and Architectures*, John Wiley and Sons, cap. 7, pp. 199–237, sep 2005.

ELSON, J., ESTRIN, D. "Time Synchronization for Wireless Sensor Networks", *Parallel and Distributed Processing Symposium, International*, v. 3, pp. 30186b, 2001. ISSN: 1530-2075.

ELSON, J., GIROD, L., ESTRIN, D. "Fine-grained network time synchronization using reference broadcasts", *SIGOPS Oper. Syst. Rev.*, v. 36, n. SI, pp. 147–163, 2002. ISSN: 0163-5980.

GANERIWAL, S., KUMAR, R., SRIVASTAVA, M. B. "Timing-sync protocol for sensor networks". In: *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys'03)*, pp. 138–149. ACM, 2003.

MARÓTI, M., KUSY, B., SIMON, G., et al. "The flooding time synchronization protocol". In: *Proc. of the 2nd international conference on Embedded networked sensor systems (SenSys'04)*, pp. 39–49. ACM, 2004.

LENZEN, C., SOMMER, P., WATTENHOFER, R. "Optimal clock synchronization in networks". In: *Proc. of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*, pp. 225–238. ACM, 2009.

GELYAN, S. N., EGHBALI, A. N., ROUSTAPOOR, L., et al. "SLTP: scalable lightweight time synchronization protocol for wireless sensor network". In: *Proceedings of the 3rd international conference on Mobile ad-hoc and sensor networks*, MSN'07, pp. 536–547. Springer-Verlag, 2007.

SOMMER, P., WATTENHOFER, R. "Gradient clock synchronization in wireless sensor networks". In: *Proc. of Int. Conf. on Information Processing in Sensor Networks (IPSN'09)*, pp. 37–48. IEEE Computer Society, 2009.

ANGLUIN, D. "Local and global properties in networks of processors (Extended Abstract)". In: *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pp. 82–93, New York, NY, USA, 1980. ACM. ISBN: 0-89791-017-6.

ARANTES, JR., G. M., FRANÇA, F. M. G., MARTINHON, C. A. "Randomized generation of acyclic orientations upon anonymous distributed systems", *J. Parallel Distrib. Comput.*, v. 69, n. 3, pp. 239–246, 2009. ISSN: 0743-7315.

MURRAY R. SPIEGEL, S. L., LIU, J. "Mathematical Handbook of Formulas and Tables". McGraw-Hill Professional, 2008. ISBN: 0071548556.

LUNDELIUS, J., LYNCH, N. "An upper and lower bound for clock synchronization", *Information and Control*, v. 62, n. 2-3, pp. 190–204, 1984. ISSN: 0019-9958.

SALLAI, J., KUSY, B., LEDECZI, A., et al. "On the Scalability of Routing Integrated Time Synchronization". In: *3rd European Workshop on Wireless Sensor Networks (EWSN 2006)*, 2006.

PUSSENTE, R. M., BARBOSA, V. C. "An algorithm for clock synchronization with the gradient property in sensor networks", *J. Parallel Distrib. Comput.*, v. 69, pp. 261–265, March 2009. ISSN: 0743-7315.

LENZEN, C., LOCHER, T., SOMMER, P., et al. "Clock Synchronization: Open Problems in Theory and Practice". In: *Proc. 36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'10)*, pp. 61–70, 2010. ISBN: 978-3-642-11265-2.

SU, W., AKYILDIZ, I. F. "Time-diffusion synchronization protocol for wireless sensor networks", *IEEE/ACM Trans. Netw.*, v. 13, n. 2, pp. 384–397, 2005.

BOULIS, A. "Castalia: A simulator for Wireless Sensor Networks and Body Area Networks - User Manual". 2010.

VARGA, A., HORNIG, R. "An overview of the OMNeT++ simulation environment". In: *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pp. 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN: 978-963-9799-20-2.

GONÇALVES, V. C. F., LIMA, P. M. V., MACULAN, N., et al. "A distributed dynamics for webgraph decontamination". In: *Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part I*, ISoLA'10, pp. 462–472. Springer-Verlag, 2010.